WL-TR-91-1014

IMAGE UNDERSTANDING RESEARCH

R. Nevatia

University of Southern California
Institute for Robotics and Intelligent Systems
Powell Hall Room 204
Los Angeles, CA 90089-0273

February 1991

Final report for period June 89 to September 90

AVIONICS DIRECTORATE
WRIGHT LABORATORY
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6543

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188
Exp Date Jun 30, 1986

1a REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY None	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b DECLASSIFICATION / DOWNGRADING SCHEDULE None	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S) WL-TR-91-1014
6a NAME OF PERFORMING ORGANIZATION University of Southern California 6b OFFICE SYMBOL (If applicable)	7a NAME OF MONITORING ORGANIZATION Avionics Directorate (WL/AAAT) Wright Laboratory
6c ADDRESS (City, State, and ZIP Code) Institute for Robotics and Intelligent Systems Powell Hall Room 204 Los Angeles, CA 90089-0273	7b ADDRESS (City, State, and ZIP Code) Wright-Patterson Air Force Base 45433-6543
8a NAME OF FUNDING / SPONSORING ORGANIZATION Wright-Patt AFB 8b OFFICE SYMBOL (If applicable) WL/AAAT-2	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F33615-87-C-1436
8c ADDRESS (City, State, and ZIP Code) 1400 Wilson Blvd. Arlington, VA 22209-2308	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. 62301E; PROJECT NO. 3119; TASK NO 00; WORK UNIT ACCESSION NO 05
11 TITLE (Include Security Classification) Image Understanding Research	
12 PERSONAL AUTHOR(S) R. Nevatia	
13a TYPE OF REPORT Final Technical Report 13b TIME COVERED FROM 6/89 TO 9/90	14 DATE OF REPORT (Year, Month, Day) February 1991 15 PAGE COUNT 125
16 SUPPLEMENTARY NOTATION	

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

This report summarizes the USC Image Understanding research projects for the period of June, 1989 to September 1990 on Contract #F33615-87-C-1436. Along with our previous annual reports (USCIRIS #238 and #258), it also constitutes our final report for the project. This report consists of a summary section, followed by a number of detailed technical papers. These papers have already been published in conference or workshop proceedings; to save time and effort, we have reproduced these papers in the final document as they originally appeared. The work in these detailed papers and in the previous reports is covered only briefly in this summary. Our research activity under this contract has focussed on the topics of 3-D vision, mapping from aerial images, and parallel processing.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL Louis Tamburino	22b TELEPHONE (Include Area Code) (513) 255-7648 22c OFFICE SYMBOL WL/AAAT

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

Louis A. Tamburino
LOUIS A. TAMBURINO
Project Engineer

Edward L. Gliatti
EDWARD L. GLIATTI, Chief
Information Processing
Technology Branch

FOR THE COMMANDER

Charles H Krueger
CHARLES H. KRUEGER
Director
System Avionics Division
Avionics Directorate

If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify WL/AAAT-2, WPAFB, OH 45433-6543 to help us maintain a current mailing list.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

Contents

1 Introduction

This report summarizes the USC Image Understanding research projects for the period of June, 1989 to September 1990 on Contract #F33615-87-C-1436. Along with our previous annual reports (USCIRIS #238 and #258), it also constitutes our final report for the project. This report consists of a summary section, followed by a number of detailed technical papers. These papers have already been published in conference or workshop proceedings; to save time and effort, we have reproduced these papers in the final document as they originally appeared. The work in these detailed papers and in the previous reports is covered only briefly in this summary.

Our research activity under this contract has focussed on the following major topics:

- 3-D Vision
- Mapping from Aerial Images and
- Parallel Processing

A summary of these areas is given in the next section.

2 Summary

2.1 3-D VISION

Our goal is to develop techniques for description and recognition of complex 3-D objects in complex scenes. We focus on the analysis of objects using shape (as opposed to color, texture or other cues) and have made significant progress in the contract period. In particular, we have concentrated on the following:

- *Range image analysis*
  We have made progress in the automatic acquisition of models from multiple views, using either symbolic or iconic representations. These models are useable for a variety of applications, including object recognition as in a system described in earlier reports [14].

- *Stereo*

  - We have completed a system which combines area-based and feature-based processing to generate dense disparity maps.
  - We have excellent results performing stereo matching using very high level primitives resulting from perceptual organization
  - In the special case of urban scenes, we have used "snakes" to accurately delineate the contours of building tops.

- *Shape from contour*
  We have developed a theory for inferring the 3-D shape of objects from their contours. This technique relies on observations of certain types of symmetries in the contours and the mathematical constraints that derive from them. Our technique uses relatively few assumptions and heuristics and is largely based on geometric properties of contours. We have shown that it is applicable to the analysis of zero-Gaussian curvatures surfaces, straight homogeneous generalized cylinders, and "snakes" and are working on extending it to yet more complex objects. Good results are obtained, however, currently we assume that contours and symmetries are given to our system. In separate projects, we are investigating the computation of such symmetries.

- *Symmetry Detection and Perceptual Grouping*
  Grouping of contours detected in an image is crucial for proper segmentation and description of objects in a scene. In our previous work, we found that symmetries play a key role in computing such perceptual groupings [44, 31]. Symmetries are also central to our technique for inferring 3-D shape from contours. In recent work, we have been investigating efficient ways of computing these symmetries [51]. Once edge contours are represented by approximating B-splines and the corners are detected [49], the computation of symmetries is of complexity $O(n^2)$, where n is the number of spline segments as opposed to the number of points.

- *Matching*
  We have defined a methodology based on efficient coding and hash tables to recognize

objects in a cluttered environment, even when the number of models is large. We can successfully recognize flat objects under affine transform, and 3-D objects given 3-D data (such as range images), with no restrictive assumptions on the shape of these objects.

2.1.1 RANGE IMAGE ANALYSIS

Range imagery differs from intensity imagery in that the input directly relates to the geometric *shape* of the objects in the scene. Our previous work has allowed us to compute symbolic descriptions of range images, and to perform matching with multi-view models. Recently, we have obtained integrated representations of models from multiple views, which is more natural since such models can be observed offline from many positions. The model building procedure is performed either by merging at the data level prior to segmentation, or by merging the segmented views, as explained below.

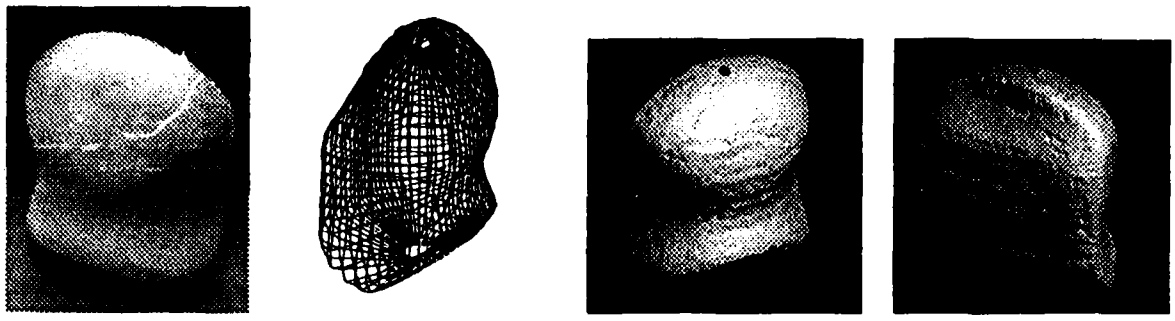
Range finders We have two different range finding systems available to generate a range map of a given 3-D object, both of them based on active triangulation. The first consists of an independent laser system generating a sheet of light projected on the target object, which is placed upon a translation or a rotary table driven by a personal computer. This computer includes a video digitizer board with two CCD cameras looking at the scene from both sides of the sheet of light. This is reported in detail in [21] and in the past annual report. This low cost system is accurate and can produce a registered intensity image of the scene along with the range image

In the case where we can not or do not wish to move the scene on a tray, we use a system that consists of a nematic liquid crystal mask inserted into a slide projector to provide an illumination pattern and a CCD camera looking at the scene from a different angle. The hardware was provided courtesy of Prof. S. Inokuchi from Osaka University, and the details of the system can be found in [52]. The main advantage of this system is speed, since by projecting a set of n Gray-coded patterns onto the scene, we obtain depths for 2^n lines.

Data level merging One of the difficulties of integrating multiple views is in finding an accurate transformation between data obtained from different views. Previous research has suggested determining the relative motion between views by using marks and regular patterns in the scene by taking intensity images at the same time and matching those features [59], or by matching surface features directly [15]. These techniques rely solely on the accuracy of feature detection and provide no feedback from the data themselves as to how well the different views have been registered under the estimated transformation.

Our approach is to use range data directly and to register successive views of the object with overlapping areas to compute transformations for the relative motion between views. To reduce the possible large search space and ensure that the algorithm converges, we assume that the approximate transformation between the data from two views is known, which is reasonable when the range data are acquired in a controlled environment.

To register two overlapping range views of the object, we first choose a set of surface points, called control points, from one range image, and then apply a minimization process to find the rigid transformation which minimizes a distance measure from those control points



(a) Original image (b) Reconstructed model (c) Rendered image of model (d) Rendered image of model

Figure 1: Object Modeling:

to the surface represented by the other range images. This minimization process is done by using an iterative least-square method. The control points and the distance measure have been chosen so that this process converges rapidly.

To merge multiple views, we use a cylindrical/spherical representation for simple compact objects. Successive range image views of the object are merged after being mapped to an object-centered coordinate system by using the relative transformations found by the registration process. To avoid the introduction of a cumulative error term in the integration process, we also use a global registration strategy, i.e., we always register the next view with the current integrated result. An example is illustrated in Figure 1 where a the wood block (a) has been viewed from 8 side positions 45° apart, from the top and the bottom. The reconstructed views of the object are shown as shaded images in (c) and (d).

Generating surface descriptions In order to obtain useful surface descriptions, we need both to devise a proper formalism using the criteria of *richness, stability* and *local support*, and also to design proper implementation tools to deal with real images (noise, quantization and digitization).

We have chosen to segment range images into simple surface patches, whose boundaries correspond to surface discontinuities (C_0) or surface orientation discontinuities (C_1). Each surface patch is then approximated globally by a bivariate quadratic polynomial [13]. This segmented representation of a scene may be viewed as a graph whose nodes capture information about the individual surface patches and whose links represent the relationships between them, such as occlusion and connectivity. Simple reasoning on these relationships is used to decompose the full graph into disjoint subgraphs corresponding to different objects. An example is shown in figure 2a-c.

The success of this representation critically depends on our ability to compute the necessary attributes, such as gradients and curvature, from an image in the presence of noise. We have found adaptive smoothing to be a tool of great value for such operations. The details can be found in [50, 49], but the ideas can be summarized as follows: *The general purpose of our Adaptive Smoothing scheme is to smooth a signal - whether it is an intensity*

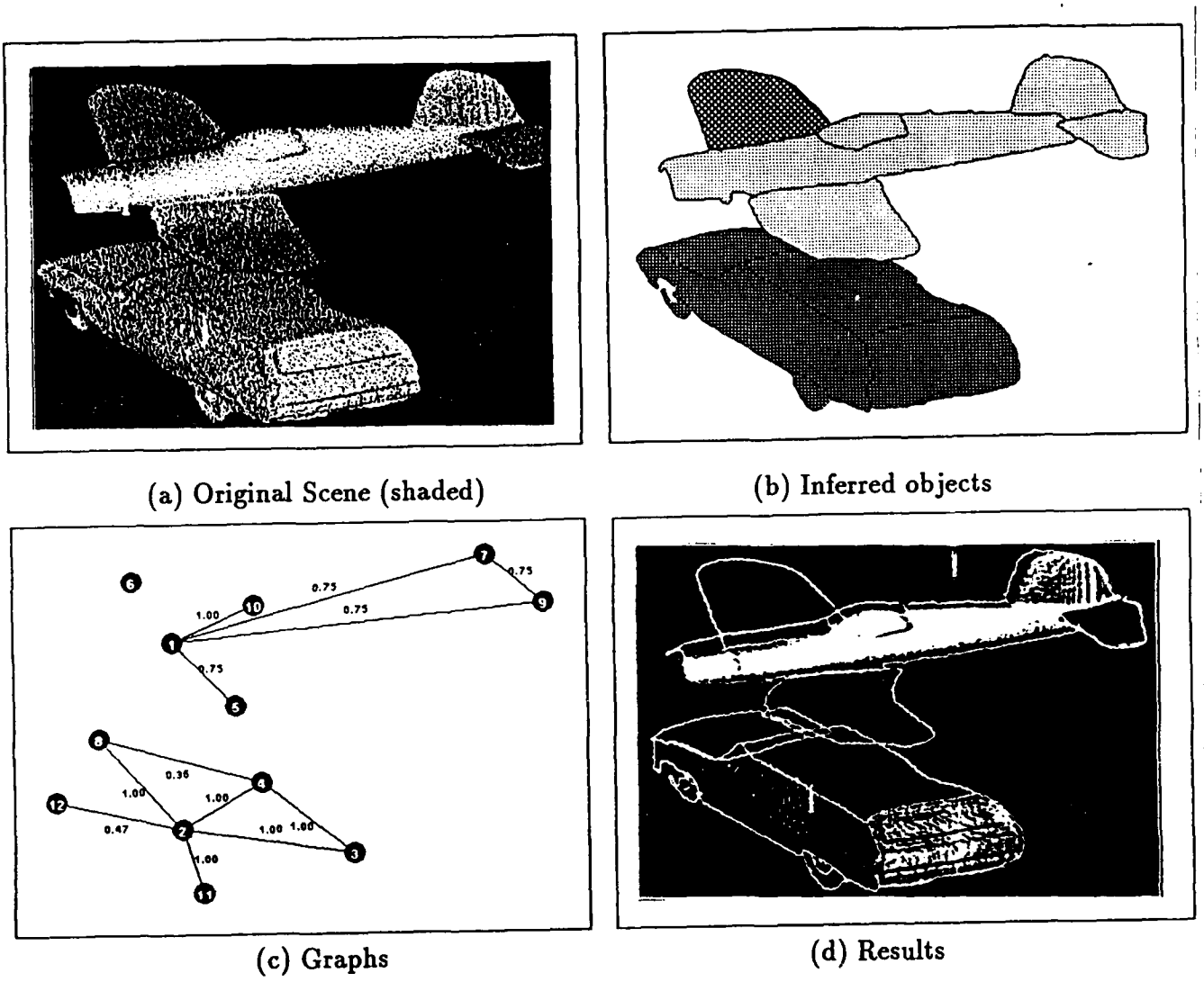
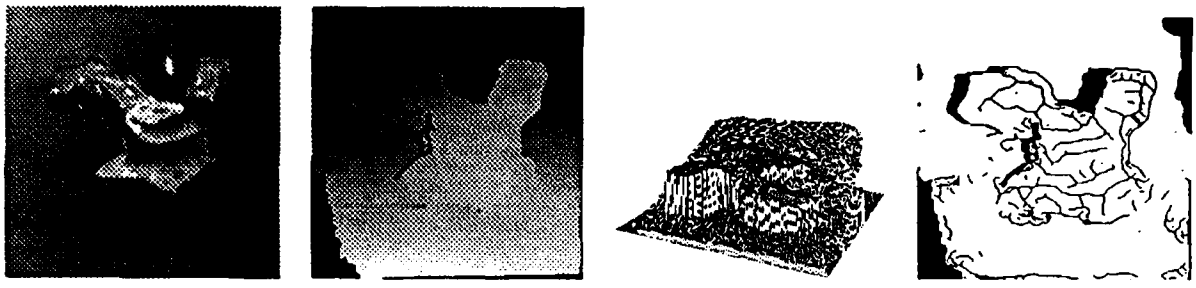


Figure 2: Segmentation of a complex range image.

image, a range image or a planar curve - while preserving and even enhancing its discontinuities. This is achieved by repeatedly convolving the signal with a very small averaging filter modulated by a measure of the signal discontinuity at each point. A relatively small number of iterations is needed to obtain a smooth signal suitable for feature extraction. In range images, we use curvature features such as curvature extrema or zero-crossings which are easily detected and directly localized after Adaptive Smoothing as opposed to Gaussian Scale-Space approaches where a tedious tracking procedure is needed.

3-D Object Recognition/Symbolic level merging We have been able to use the above descriptions to achieve successful recognition of complex objects in scenes containing multiple objects that are only partially visible and are occluding each other. An example of recognition is presented in figure 2(d), and a detailed treatment can be found in [12, 14]. For the purpose of matching, a model is represented by a set of similar descriptions from multiple viewing angles, typically 4 to 6. Models can therefore be acquired and represented automatically. Matching between objects in a scene and models is performed by three modules: the *screener*, which finds the most likely candidate views for each object, the *graph*



(a) Intensity (b) Disparity (c) Disparity (d) Features

Figure 3: Renault Part.

matcher, which performs a detailed comparison between the potential matching graphs and computes the 3-D transformation between them, and the *analyzer*, which takes a critical look at the results and proposes links to split and merge object graphs.

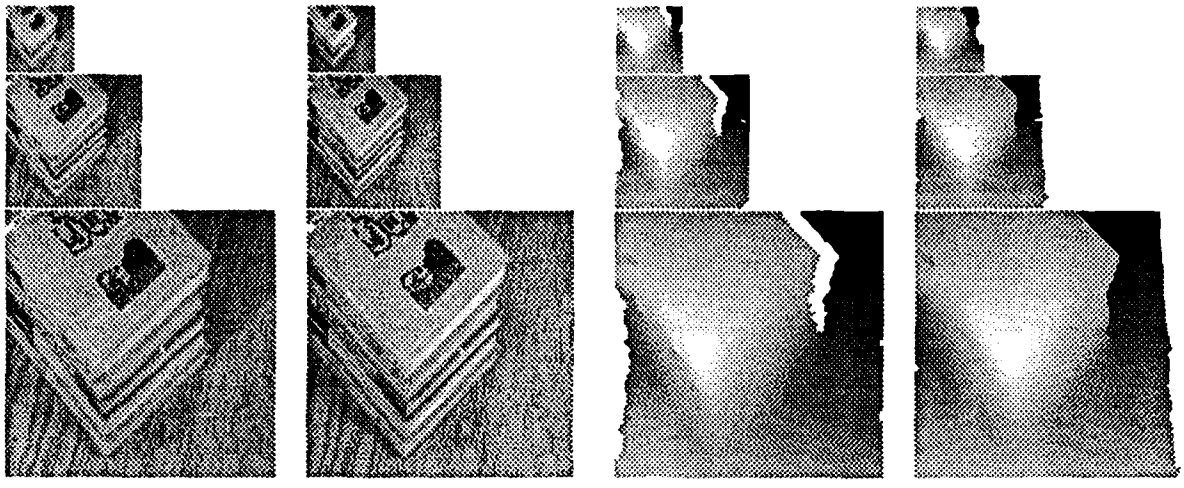
The alternative approach consists of generating a symbolic description, such as an attributed graph, for each view, and then merge the different descriptions at this high level. Each view is represented by a graph whose *nodes* are the individual surface patches and the *links* are the relationships between adjacent patches. The matching between views is achieved either through a tree search procedure [14], or by a 2-level constraint satisfaction network [40]. One of the difficulties to be overcome by this process is the inference of surface patches from bounding contours, since these are not necessarily continuous and generally inaccurate at junctions. We have obtained good results by modeling this process as a dynamic network subject to *weak smoothness* constraints. The initial state of the network consists of the curves produced by low level operators, but these decay over time unless excited. Possible completions provide this excitation, competing with each other and strengthening existing curves [41].

2.1.2 STEREO

We are using different approaches to solving the stereo correspondence problem, including using a combination of area-based and feature-based processing, and working with complex primitives resulting from a perceptual grouping process. We also are using active contours to obtain accurate boundaries of roof tops in aerial views of urban areas.

Feature and area-based processing We have considerably improved the system described in earlier reports [9], which integrates area-based and feature-based processing, by taking advantage of the unique attributes provided by each one separately. The area-based processing generates a dense disparity map, and the feature-based processing accurately locates discontinuities. The first improvement, described in [10], is the extraction of depth and, in many cases, orientation discontinuities from the image.

Figure 3 shows the results obtained for the "Renault Part" stereo pair. Figure 3 (a) and (b) show one of the stereo intensity images and the respective disparity result; (c) shows a 3-D plot of the disparity, from which the surface features (d) were extracted. The surface features located on the disparity surface are the depth discontinuities, the occluded regions,



(a) Intensity (b) Disparity

Figure 4: Books – Multi-resolution Pyramid.

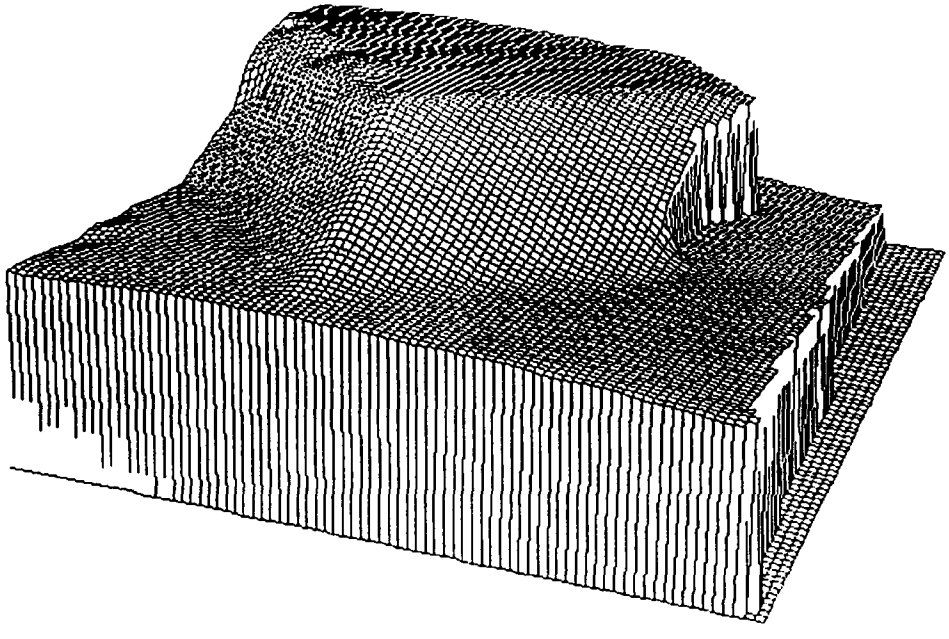


Figure 5: Books — 3-D Plot of Disparity.

and the concave and convex folds.

The second improvement is the use of a multi-level pyramid, first processing a reduced (coarse) version of the image pair, and then propagating the results to another level for higher-resolution (finer) processing, as shown in figure 4. This introduces a more global context and allows the correction of local errors in matching, such as those due to photometric and geometric distortions. Figure 5 shows a 3-D plot of the disparity and figure 6 shows the extracted surface features. We have applied this Stereo Vision System to a wide variety

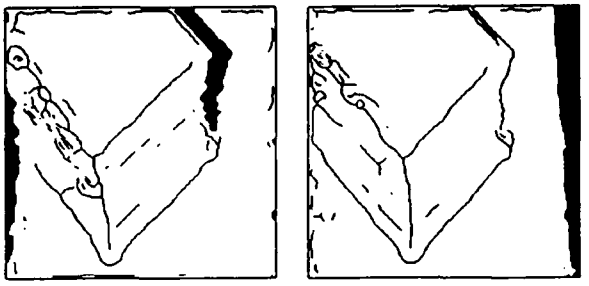
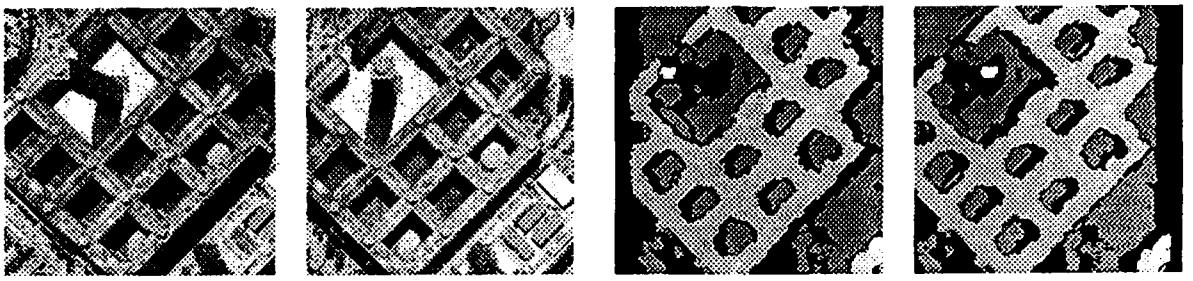


Figure 6: Books — Surface Features.



(a) Intensity Images (b) Disparity Images

Figure 7: Jussieu

of scenes and obtained results which compare very favorably with state-of-the-art methods [39, 18, 11].

Stereo of aerial urban scenes Current stereo algorithms, whether area-based or feature-based, tend to fail around depth discontinuities, since these are the locations where smoothness assumptions do not hold. This phenomenon is most easily observable in aerial views of urban scenes, where the roofs of buildings can be detected, but not accurately delineated. Fua [16] and Mohan [34] propose to solve the problem by restricting the possible shapes in the form of a generic model.

Here instead, we propose to use the initial estimate provided by a traditional stereo system (as described in the last section). and to refine it by enforcing a local smoothness constraint. This is accomplished by an active contour model, whose details are given in this report [30]. The estimate is shown in figures 7–9. We have obtained excellent results, even when the boundaries contain corners, as illustrated on figure 10.

Stereo matching using high level features We are also investigating an alternative approach to stereo that uses high level features for correspondence. Lower level feature matching may have difficulties with global correspondence, particularly when repetitive structures are present, requires presence of rather dense texture and highly accurate knowledge of epipolar geometry. High level feature matching can potentially overcome these obstacles. Further high level features are fewer in number and hence should be faster to match. However, this approach has the deficiency that high level features need to be computed from monocular images; a process that is known to be difficult and error prone. We have developed

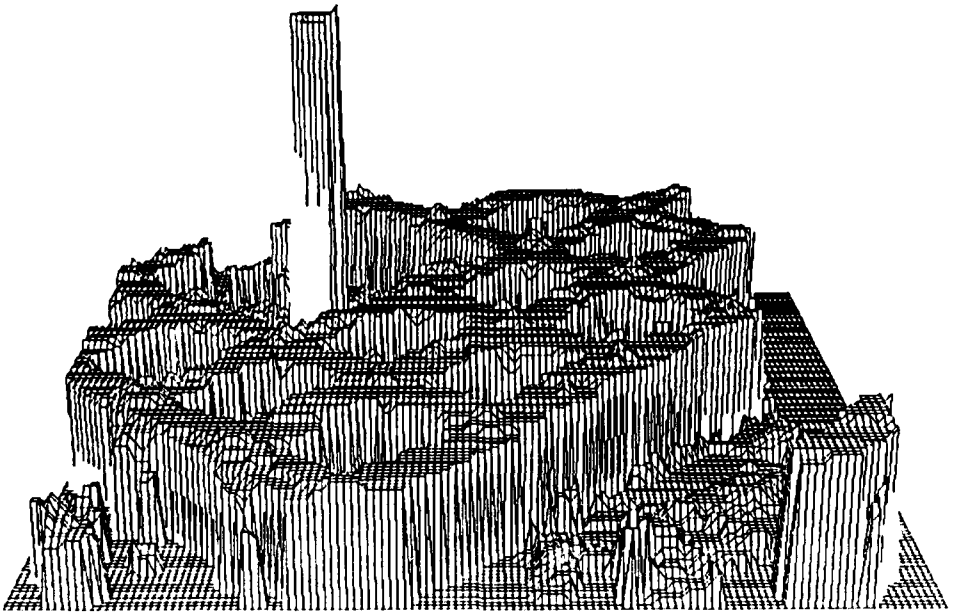


Figure 8: Jussieu — 3-D Plot of Disparity.

Figure 9: Jussieu — Surface Features.

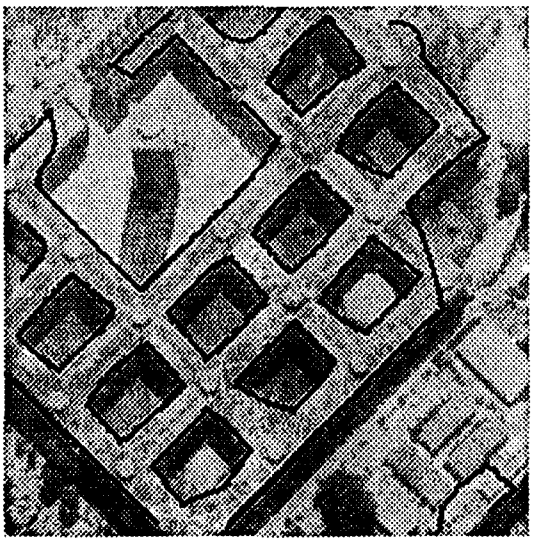
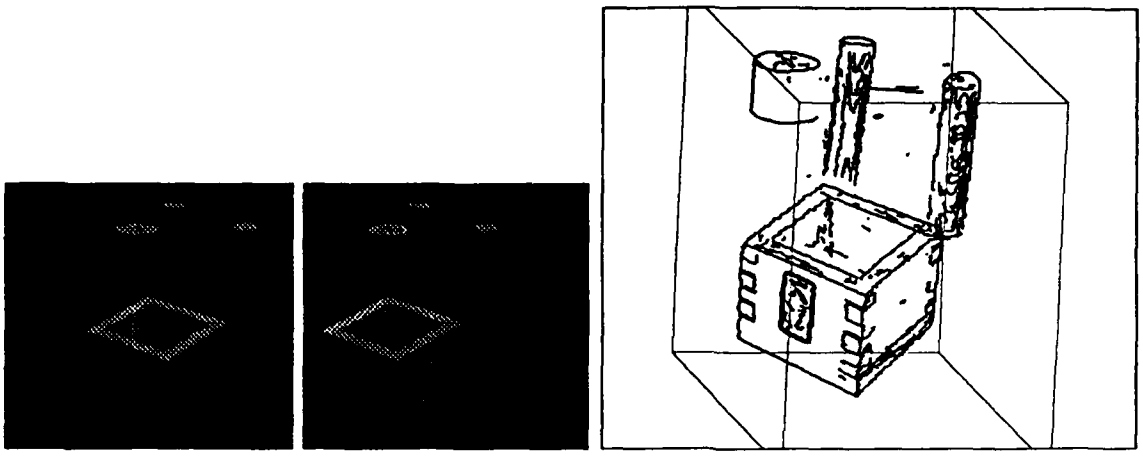


Figure 10: Example of delineation of buildings roofs with deformable contour models



(a) Left image (b) Right image (c) Disparity output

Figure 11: Results of a scene with multiple occlusions

sophisticated perceptual grouping methods to overcome this difficulty [31].

Our first experience with high level stereo was in the context of analyzing buildings in aerial scenes. In such scenes, texture (on the roofs) is very sparse and disparity changes discontinuously at the boundaries. We found that using high level features (rectangles) was very effective for stereo processing of such scenes [34]. We next investigated generalization of this approach to scenes where the object shape is not so constrained [34]. In this work, we found that ribbons (defined by two symmetrical curves with closures at the two ends) are an effective method for organizing the curves in an image into higher level features and that these ribbons could be used for stereo matching. This work, however, concentrated on the grouping problem and not on development of a competent stereo system.

In our recent work, we have been building on our perceptual grouping system to develop a stereo system. Features such as edgels, curves, symmetries, and ribbons which represent geometric structures of objects in the scene are extracted from each image using perceptual grouping. The grouping algorithms are similar to those described in [34] but several enhancements have been incorporated. A hierarchy of features from the left and right images are matched using a relaxation network. Our method has shown accurate results for images with multiple occlusions and wide angle disparities. Results from this method are illustrated in figure 11.

2.1.3 3-D SHAPE FROM CONTOURS

Humans are able to readily perceive 3-D shape from a monocular image. Many cues are used in this process such as shading, shadows and texture. However, we believe that the most significant cue is the shape of the 2-D contours. The process of inferring 3-D shape from contours, however, has proven to be a very difficult one. We believe that we have made a major advance in this area and have developed a theory that significantly extends the range of shapes that can be analyzed. Our theory relies on observations of symmetries in the scene and the conjecture that only shapes having certain symmetries are percieved in

3-D by humans.

We define two types of symmetries that we call *parallel* and *mirror* symmetries (the precise definitions are given in another paper in this report [58]. Given the observations of these symmetries in some specific combination, we can infer some qualitative properties of surfaces and objects in the scene, such as whether they are planar, have a zero-Gaussian curvature surface, or are some specific classes of generalized cylinders.

Further, the contours and the symmetries allow us to formulate some constraints on the quantitative shape of the surfaces being viewed. The constraints that derive purely from the geometry of the surface are, however, not sufficient to compute the precise shape of the surface and leave some degrees of freedom unconstrained. These degrees of freedom can also be fixed by using some simple perceptual properties.

Our technique is rather mathematical and hence difficult to summarize without introducing a good deal of notation. Hence, we will only give references to the more detailed work and show some examples. The basics of our method, and its applications to analysis of zero-Gaussian curvature surfaces are given in [57]. Figure 12 shows some examples from this work. The first column of this figure shows the input contours to the program, the middle column shows the computed surface orientations as a "needle diagram" and the last column shows the surface orientations by painting the surface with intensities that would result from a Lambertian surface illuminated by a point source. Extensions of our method to straight homogeneous generalized cylinders (SHGCs) and snakes (generalized cylinders of constant cross-section) and some results are included later in this report [58].

We hope that these examples indicate the power and range of our approach. We are in the process of further developing the theory to apply to yet more complex objects. It should be noted that this technique assumes that the appropriate contours and symmetries are given; this is far from a trivial task. However, we are making progress on detection of the appropriate symmetries in other projects in our group [33, 51].

2.1.4 SYMMETRY DETECTION

Once edges are extracted, the resulting contours must be represented for further reasoning. Iconic representations do not make the necessary information explicit: by definition edgels only capture very local properties of an image, and the inference of higher structures, such as object boundaries, requires *grouping* operations. We believe that such operations rely on basic and simple properties and various forms of symmetry [31]. The representation must therefore make explicit differential properties of contours, such as tangent and curvature. Furthermore, because of the variability inherent in the imaging process, the representation should tolerate noise, partial occlusion, and perspective, thus suggesting segmented, local descriptors [45].

If the world was composed of polyhedral objects alone, we would know to expect only straight line segments in images, and polygonal approximations would be appropriate. In many cases, such an approximation is indeed sufficient, as demonstrated by several applications such as stereo [29], aerial image understanding [20] or object recognition [35, 53], but is unable to capture curvature information, since it is a first order approximation. Also, if a

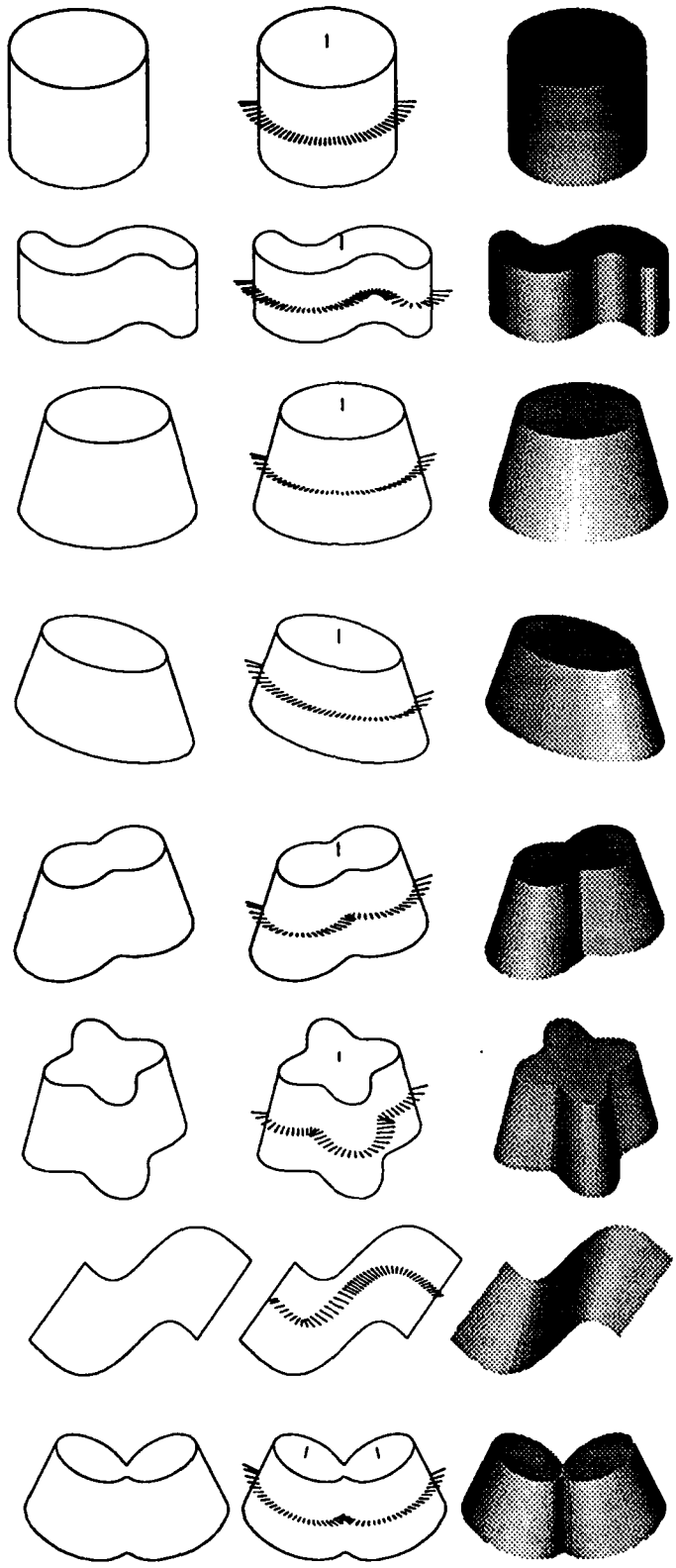
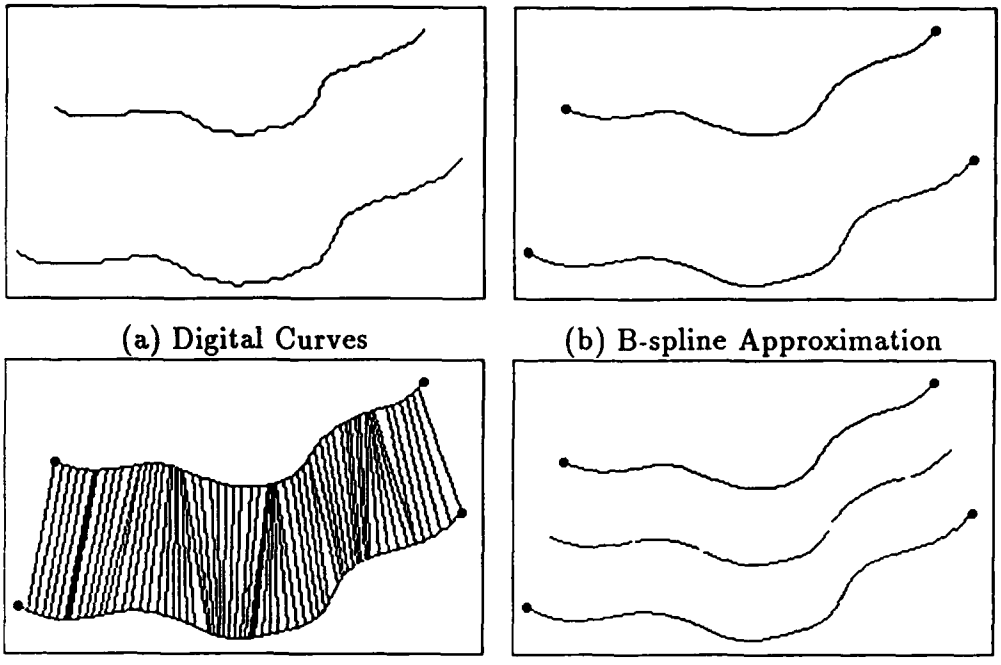


Figure 12: Sample contours, the computed needle images and the images after shading the object using the computed orientation at all surface points. The last object has a non-planar cross section and was segmented into two planar cross section objects before processing.



(a) Digital Curves (b) B-spline Approximation

(c) Lines of Symmetry (d) Symmetry Axis

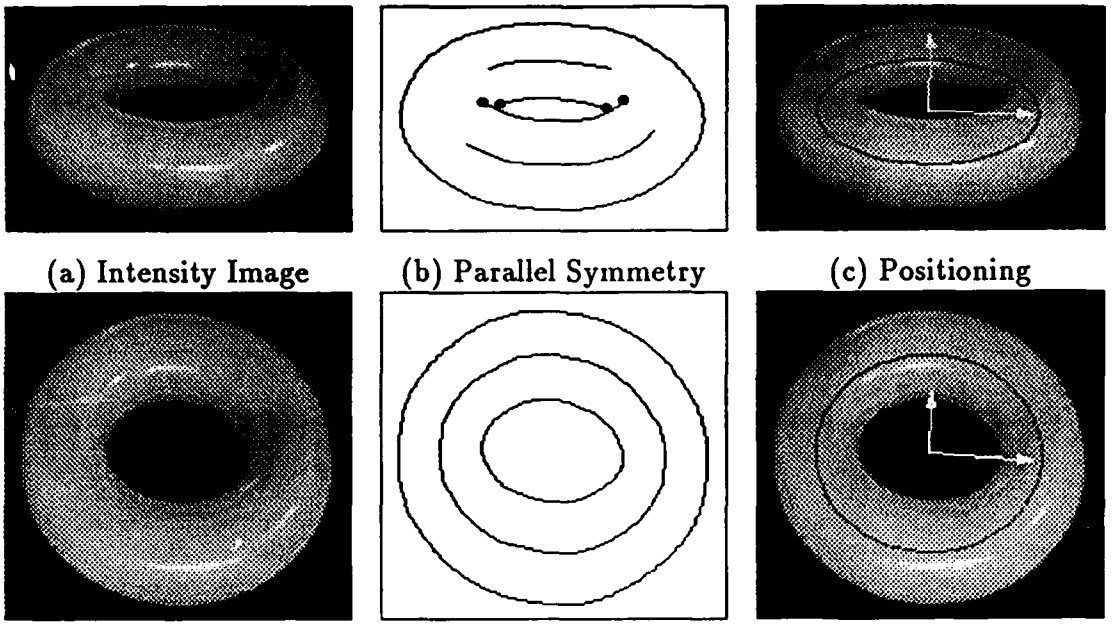
Figure 13: Detection of Elementary Parallel Symmetries

contour is smooth, the number of points required to approximate it may be quite large, and the exact position of the points somewhat unrelated to the contour itself. These issues have been tackled by the graphics community in the context of design, and we propose to use some of the resulting tools, particularly approximating B-splines. The resulting representation is compact and faithful to the original data for smooth or piecewise smooth contours, open or closed.

It is also very well suited for the detection of symmetries. While it is easy to define symmetry between two infinite straight lines, the concept of symmetry between curves is harder to define: Rosenfeld [47] provides a lucid account of the differences between Blum's [6], Brooks' [7], and Brady's [5] definitions, and a more recent paper by Ponce [42] gives further comparisons. Here, we are interested not in local symmetries which provide skeletal shape primitives, but rather in symmetries which help to infer shape from contour: Nevatia and Ulupinar [56] postulate that they are skewed and parallel.

These can be computed efficiently using our B-spline representation. The main advantages are the low computational complexity ($O(n^2)$, where n is the number of spline segments instead of the number of points) of the process and the stability of the results. Figure 13 shows an example of parallel symmetry detection using a quadratic B-spline approximation starting from the two digital curves displayed in figure 13(a).

As an application, for the very specific case of a torus, the detection of parallel symmetries allows us to infer the 3-D orientation of the object in a much simpler fashion than proposed in [43], as shown on figure 14.



(a) Intensity Image (b) Parallel Symmetry (c) Positioning

(d) Intensity Image (e) Parallel Symmetry (f) Positioning

Figure 14: Positioning of a Torus

2.1.5 MATCHING

Object recognition involves identifying a correspondence between part of an image and a particular view of a known object. This requires matching the image against stored object models to determine if any of the models could produce a portion of the image. We have actively promoted the idea that higher level features organized in graphs are the key to recognition in the presence of occlusion and photometric variations [14, 28, 37]. Recently, we have addressed the issues involved in recognizing objects in a cluttered environment when the number of models is large. We have been able to show excellent results for the recognition of flat objects under affine transform [53], and, in a paper later in this report, of 3-D objects given 3-D data [54]. The keys to our approach are

- a redundant representation
- Gray code to measure semantic difference
- hash tables for fast retrieval
- automatic acquisition of models

For the problem of recognition of multiple flat objects in a cluttered environment from an arbitrary viewpoint [53], the models are acquired automatically and initially approximated by polygons with multiple line tolerances for robustness. Groups of consecutive linear segments (super segments) are then quantized with a Gray code and entered into a hash table. This provides the essential mechanism for indexing and fast retrieval. Once the data base

of all models is built, the recognition proceeds by segmenting the scene into a polygonal approximation; the Gray code for each super segment retrieves model hypotheses from the hash table. Hypotheses are clustered if they are mutually consistent, and represent the instance of a model. Finally, the estimate of the transformation is refined. This methodology allows us to recognize models in the presence of noise, occlusion, scale, rotation, translation and weak perspective. Unlike most of the current systems, its complexity grows as $O(kN)$ when N is the number of models, and $k \ll 1$. An example of successful recognition is shown in figure 17 in the aerial image section of this introduction.

For the recognition of 3-D objects from 3-D data, we use a data structure called a *splash*, which describes the variation of surface normals in a circular neighborhood of a point, encoded as a super segment. From then on, the matching methodology is identical to the 2-D case. The full details can be found later in this report [54].

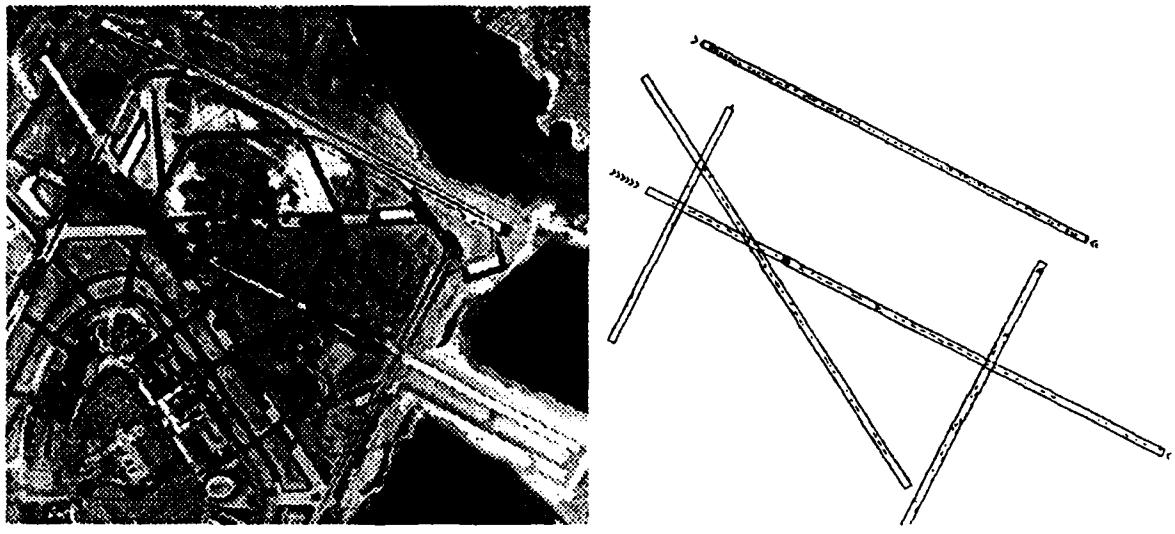
2.2 AERIAL IMAGE ANALYSIS

We have three projects for the analysis of images of aerial scenes including efforts to develop modules that exhibit high performance by themselves, the integration of modules into systems, and the formulation of a theory to define the underlying "visual abilities" required and useful for extraction of cultural features from images of aerial scenes:

- The focus of our work in the past has been the development of modules for detection and description of cultural (man-made) features present in aerial scenes such as the transportation network (fig. 15a,b) [19], building structures (fig. 16a,b,c) [20, 32, 31] and aircraft (fig. 17a,b,c) [53]. In the past report we gave a detailed example of the analysis on an airport complex. Later in this introductions, we will give an example of a module for pier and ship detection from an image of a harbor complex.

  These modules typically rely on the perceptual grouping of primitive geometric features (lines, anti-parallels, junctions, portions of rectangles, etc.) extracted from the images, to detect the objects. Modules for mobile objects such as aircraft and ships on the other hand, use models and rely on scale and rotation invariant matching techniques to detect the objects. Our current work on 2-D and 3-D matching techniques is covered in detail in [53, 54]. Typically these methods are applied at a stage where we have a great deal of confidence that these objects are (or should be) present in the image. For instance, after detection of runways, taxiways, and buildings, we can then look for aircraft in the appropriate areas. These in turn, help reinforce the runway and taxiway hypotheses as well as help determine the funtionality of some of the buildings.

- A second portion of our work has concentrated on devising a system that manages the modules and integrates the results of the modules thus providing local and global context as well as higher level reasoning suitable for the description of an entire complex or scene. In the past we have concentrated in the domain of large commercial airports, and developed modules for detecting major structures. Now we are investigating the interaction of these modules. We hope to report on this work in a future paper.



(a) JFK Airport (b) JFK Runways

Figure 15: Runway detection module

- Our third project concentrates on the development of general techiques. These include devising a taxonomy of perceptual grouping operations and, the development of a language for describing tasks in terms of grouping operations. We expand on these topics in the following sections.

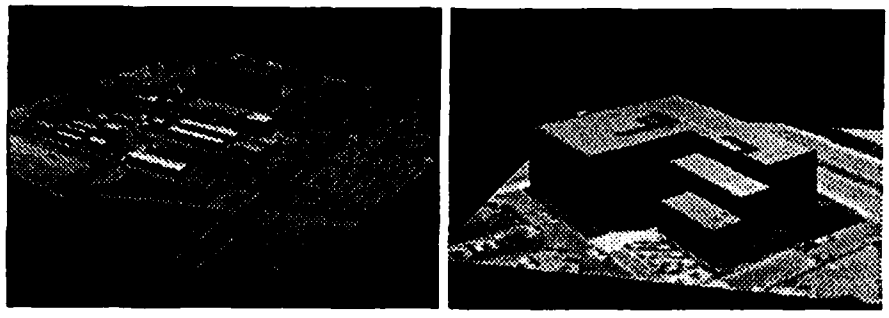
2.2.1 DEVELOPMENT OF GENERAL TECHNIQUES

We believe that a hierarchy of processing steps is the appropriate approach for aerial image understanding, where the levels of the hierarchy are chiefly determined by three factors:

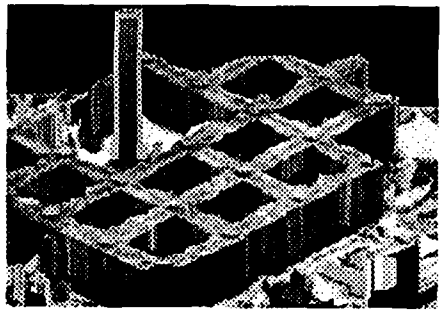
1. The available sources of knowledge, both generic and domain specific. We know for instance that airport runways are straight (geometry), and that they must have standard markings (object specific) applied to the surfaces for safety and to aid pilots.

2. The available image resolution and quality. For example, it is more desirable to look for global features, such as harbor piers, at lower resolutions and then apply the model-to-feature matching to small portions of high resolution images to locate the ships (see below). Why? Because the pier areas are salient features, a collection of macro features arranged in some simple geometric fashion along the boundary of two distinct regions, land and water. The detection of ships, and perhaps their classification by type on the other hand, requires higher resolution and more symbolic processing.

3. Measurements and assertions as a function of scale. What can or should be measured at a given scale? Invariably we can get bogged down by considering everything possible at all scales, and build complex and massive data structures. However, this is often unreasonable for mapping and photointerpretation tasks where the image content and typical resolutions quickly make such approaches unfeasible.



(a) Simple groupings (b) Complex groupings



(c) Feature/Area stereo

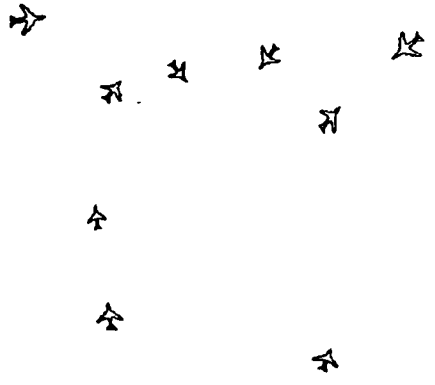
Figure 16: Building detection modules

The characterization of such hierarchies is the focus of our work and involves two fundamental issues: The development of a formal language to describe mapping and photointerpretation (or other) tasks, and the development of a grouping theory to define the generic "visual abilities" required to accomplish these tasks. We believe that many of these visual abilities can be expressed in terms of generalized classes of perceptual grouping operations that can be applied in parallel. Eventually the task descriptions should be given in terms of (or, compiled into) a sequence of alternating abstractions in the representation of the features and application of classes of grouping operations. We explore some of these ideas below using as an example the task to "detect pier areas and ships" from an aerial image of a portion of a harbor scene.

Most of this work would fit at the "middle-level" level of perception. The "connection" with the lower levels of processing, is reflected by the fact that the grouping processes are more "non-purposive," and thus should be implemented to run in parallel. The connection to higher levels of processing (reasoning about segmented objects, where an object is a single, functionally identifiable 3-D object, as determined by the task at hand) is reflected by grouping processes that are more purposive, operate on increasingly abstract features, and are sequential in nature.

For a number of years our group has developed methods and techniques involving perceptual organization. Groupings of near, parallel, collinear, co-curvilinear, and symmetric features have been used to represent, segment and extract parts or whole objects from aerial images and images of office scenes. For a reference on our most recent work see [31].

(a) LAX Airport (b) Canny Edges



(c) Detected Aircraft

Figure 17: 2-D Matcher applied to image edges for aircraft detection

Recently we have begun work towards the development of a taxonomy for grouping operations, and here we only introduce informally the notion of *grouping fields*, a general tool for describing mathematically the visual abilities that involve perceptual groupings of visual primitives closer to the lower and middle levels of perception. These are analogous to the ability that humans have to, presumably preattentively, acquire sensations that capture fundamental and basic geometric arrangements of image elements in a reflexive manner.

Briefly, the notion of a *grouping field* is analogous to force fields in nature. When a visual feature, due to its size, shape, or other property induces a perceptual grouping with other features in the field of view, we say that a *grouping field* exists around it. Conversely, any visual feature in the field of view generates a grouping field which is a function of the feature properties and can be influenced by the task at hand.

We believe that grouping fields will be useful in dealing with many of the problems pointed out in previous work by [25, 26, 27, 55, 60] and others, that attempted to derive computational approaches to perceptual organization abilities.

The combinatorial explosions that arise in attempting to establish relationships among low level features purely on the basis of attribute processing is a major problem. For photointerpretation tasks, at least, it seems that the way to avoid this is to explore the generality

aspects top-down, that is, by describing what we want, say detect piers and ships, and with our own experienced knowledge of piers and ships, generate a task description that includes the perception landmarks (first, detect border between land and water region, then detect pier areas, next detect ships in the neighborhood of pier areas, last identify ships).

2.2.2 AN EXAMPLE: ANALYSIS OF HARBOR COMPLEXES

In analyzing a harbor complex we want to be able to describe the buildings in the port facility, the transportation network around the facilities, and of course the pier areas and the ships in the area. In our example we concentrate on the piers and ships and the grouping fields and grouping operations that lead to the detection of the pier areas. We then briefly discuss ship detection and classification.

What do we need to know about port and harbor facilities to detect the piers and describe the ships? That the planning and design of port and harbor facilities is strongly dependent on the characteristics of the ships to be served and the type of cargo to be handled [61]. To eventually describe the scene completely we would need information about the ships: Main dimensions (length, beam, draft), cargo-carrying capacity, cargo-handling gear, types of cargo units, shape, hull strength and motion characteristics, mooring equipment, maneuverability, and so on.

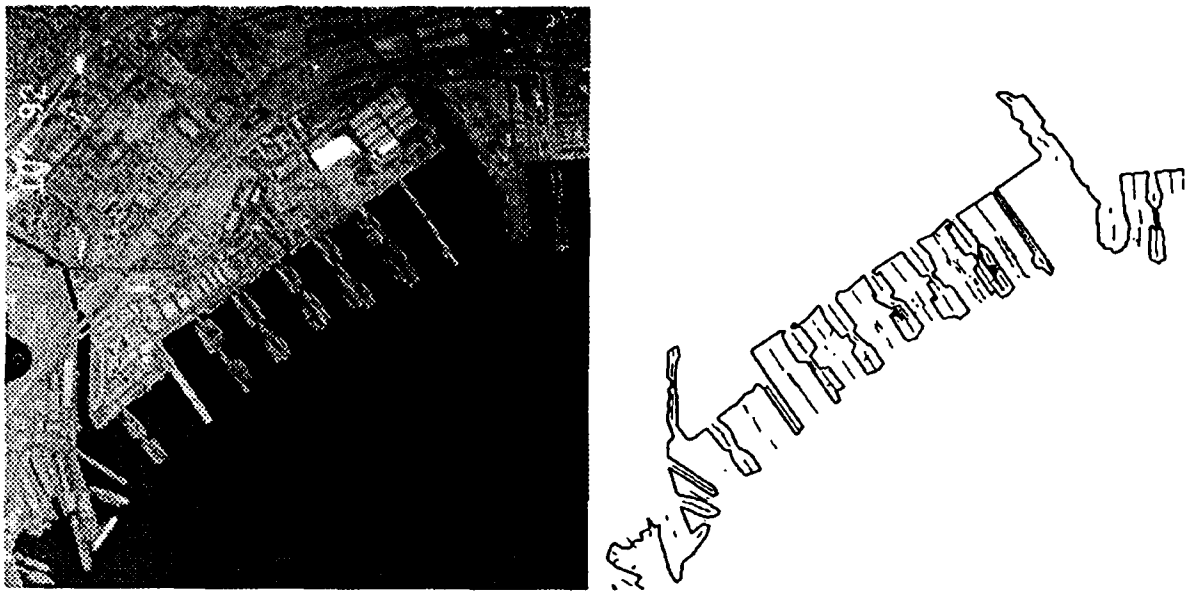
To detect only the pier areas (where later we look for ships) we only need the upper bounds on ship dimensions and the image resolution. These parameters are easily available a-priori and chiefly determine the extent and strength of the grouping fields associated with the features. Let us define some grouping classes useful for this task:

- Proximity-0D (Px0D): Groups nearby features without regard for the dimensions of the features. Each feature, whether a dot, a line, a ship, or another suitable group, generates a grouping field about its center of mass. The extent of the field (typically, circularly symetric) is determined by the field of view or by the task as a function of image resolution. Intersecting fields form a group with the same extent and a new center of mass. The strength of the field is proportional to the "mass" (a function of the complexity of the feature), and inversely proportional to the square of the distance from the feature's center of mass. Values are scaled according to resolution so that the same two features at two different resolutions attract each other with the same force.

- Proximity-1D (Px1D): Groups nearby features where a 1D attribute is dominant and can be used to constrain membership. The strength of the field in this case would be proportional to, and a function of the attribute.

- Proximity-ND (PxND): Groups nearby features with ND attributes. Each attribute requires one layer.

- Parallelism with overlap (PlwO): Groups features that are parallel with respect to their dominant orientations. Each allowed orientation determines a layer where the fields of each feature having that orientation is active. For each orientation, intersecting fields give all the features parallel to a given feature. The fields themselves have



(a) Image (b) Segments and Apars

Figure 18: U.S. Navy Facility (512x512 image)

an elliptical shape with its minor axis equivalent to the length of the feature in the dominant orientation, and its major axis equivalent to the extent of the field of view or, constrained by the task. Note that allowing for angle tolerances is equivalent to the union of fields across field layers.

- **Parallelism with no overlap (PlwnO):** The same as above with circularly symmetric fields.

- Collinearity-0D (Co0D): Groups three or more features without regard for the spatial extent of the feature. Any two of the three features determine the extent of the grouping field, typically an ellipse with high eccentricity, centered about the center of mass of the feature. The eccentricity determines the allowed tolerance in collinearity, and the extent of the field is equivalent to the extent of the field of view. The orientation of the two selected features determines a layer for field intersection. The steps in a ladder have Co0D.

- Collinearity-1D (Co1D): Groups two or more features with respect to their dominant orientation. Each feature determines the extent its GF, also an ellipse with high eccentricity. The eccentricity determines the allowed tolerance in collinearity, and the extent of the field is equivalent to the extent of the field of view, or constrained by the task at hand. The orientation of each feature determines a layer for field intersection. The fragments of an airport runway have Co1D.

Let us now apply two of these definitions to our pier example. Figure 18(a) shows an image of a portion of the U.S. Navy facilities in San Diego. We expect to see mostly military

ships that may require long term docking, thus allowing for double or triple docking. We know the image resolution and the approximate ship dimensions, thus we know the minimum size of the piers. The following gives the levels of the desired task:

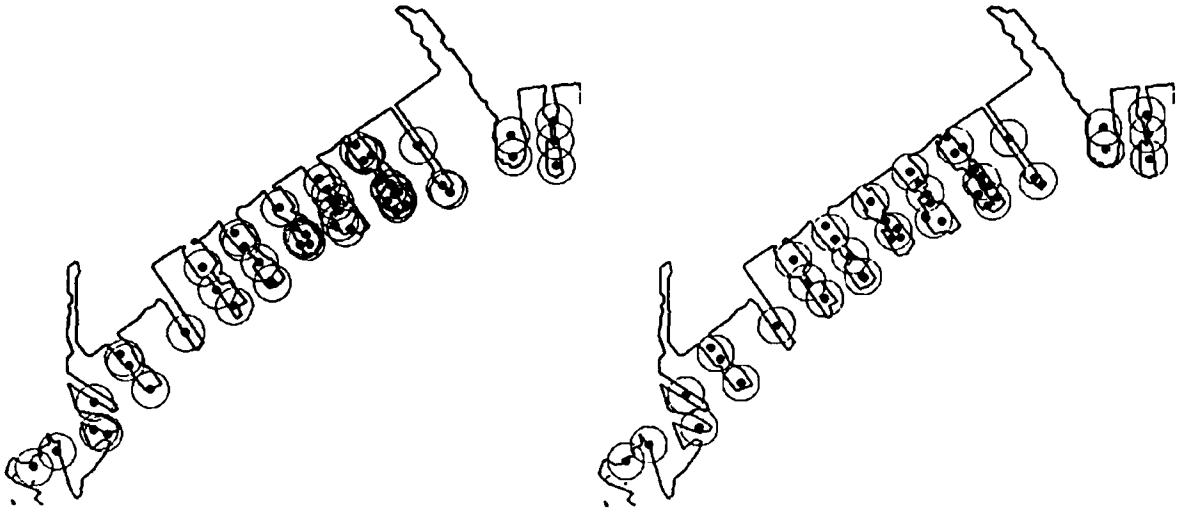
```
0: Analyze Harbor Scene.
 1: Detect and classify buildings.
 1: Detect and classify access roads.
 1: Detect and classify ships.
  2: Detect ships.
  2: Locate ship repair/construction areas.
   3: Locate ships.
    4: Classify ships.
  2: Locate Pier areas.
   3: Locate boundary between land and water.
   3: Locate ''land'' structures in water.
   3: Detect pier areas.
   3: Locate ships.
    4: Classify ships.
   3: Describe ships
  2: Describe piers.
 1: Describe piers and ships by class.
0: Describe harbor scene.
```

We now describe the task at level 2, Locate Pier Areas:

Locate Boundary between Land and Water: We detect the boundary between land and water regions automatically using a region-based segmentation procedure [38]. In this example we arbitrarily selected the largest region to represent the water region. Next we approximate these boundary by piecewise linear segments (thick lines in fig. 18(b)) using LINEAR [36].

Locate "land" Structures in Water: Contrary to many natural structures on the shores, man-made structures appear highly geometric. We expect that most piers appear as linear structures attached to the shore, and surrounded by water. Their linearity indicates that the piers or portions of piers should be characterized by anti-parallel pairs of segments of opposing contrast [36], or *apars* for short. Ships are typically docked parallel and adjacent to the piers. We then expect that most of the line segments corresponding to sides of piers, sides of ships, shadows, and so on in the neighborhood of the piers would result in many apars. The constraint on the range of separations between pair of segments (equivalent to the width of the resulting apar) is a function of image resolution and ship dimensions. The apars in our example are shown as thin lines in fig. 18(b) obtained using LINEAR.

Detect Pier Areas: The apars are easily classified into land or water using the detected water region. Subsequent processing operates on the land apars only. Next, we apply PxOD grouping. The extent of the fields is task-dependent and does not have to be precisely determined. At the resolution in our example (about 8 meters per pixel), the field's radii is roughly equivalent to a pier width plus the width of three destroyers on both sides of the piers, or about 16 pixels.



(a) PxOD - Proximity grouping (b) Selected proximity groups

Figure 19: Selected proximity groups

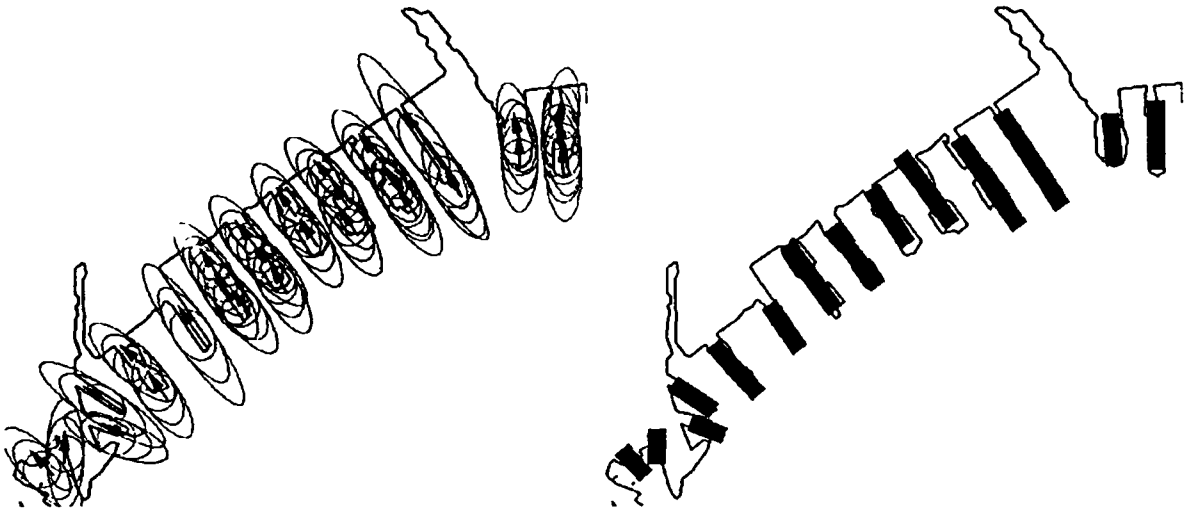
These fields (fig. 19(a)) occupy a single layer. Each field intersection operation shifts the center of mass of the group, however the field associated with the group has the same properties as the individual apar fields. We then select the groups so that apar membership is exclusive by extracting the groups in order of decreasing mass (number of apars). The resulting groups (fields) represent potential pier fragments (fig. 19(b).)

At any resolution, we expect that the lines are fragmented and incomplete, due to inefficiency in the line detection process or real structures in the image. Thus, we expect that the resulting groups represent pier fragments rather than complete pier areas. Since we expect the pier sections to be straight, the next step calls for collinearity grouping to join possible fragmented pier areas. Note that the groups in fig. 19(b) are easily perceived as being collinear.

We choose to represent the groups of apars by apars as well, having a length and width equal to the diameter of the final field. The orientation of the apar is given by the dominant orientation (the largest peak in the length-weighted histogram of the orientation) of the apars in the group (see the arrows in fig 20(a).)

Next we apply Co1D to the pier area fragments. The longest piers are about three times the length of a destroyer thus we allow the extent of the elliptic fields (see fig. 20(a)) to be up to three times the apars, and have a width equivalent to the apar width (or group radius).

The result of the grouping is then represented, again by apars, which in turn represent potential pier areas (see fig. 20(b)). These are described by their approximate length and position, and are used to extract image windows from a high resolution image of the scene where we look for ships.



(a) Potential Fragments and Co1D fields (b) Detected Pier Area

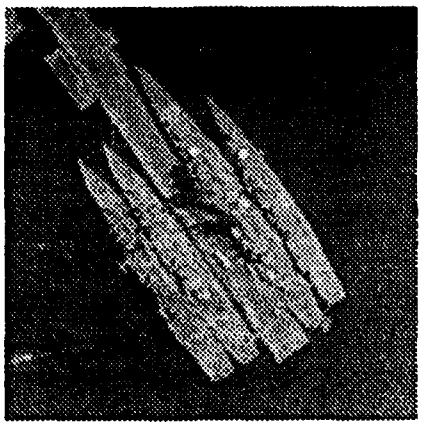
Figure 20: Pier Detection Fragments and Low Resolution Results

Locate ships: We have performed preliminary experiments to detect the ships in high resolution windows using the same matching technique that we used to detect aircraft [54]. One of these windows is shown in figure 21a with the adaptively smoothed [8] boundaries shown in figure 21b. Three coarse-to-fine models of a single and a double destroyer group were matched against these edges to obtain the detected ships in figures 21c,d.

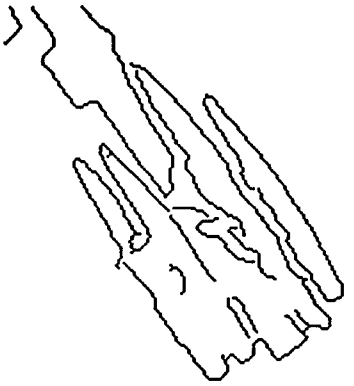
Classify ships: We consider our ship detection results preliminary. The simplicity of the ship's shape is a disadvantage for the matching technique. The double ship configurations are easier to match for the same reason. For ship identification better ship boundaries are required. We plan to apply a technique for boundary refinement using B-snakes [30]. The matching technique can then be applied with finer models for more accurate ship classification. Other alternatives for ship detection include stereo processing of these high resolution windows with an area/feature based technique [9], also followed by boundary refinement and 2D matching.

2.3 PARALLEL PROCESSING

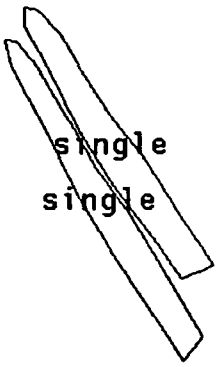
As shown in the previous sections, we are making good progress in solving some difficult image understanding problems. However, one major obstacle remains in applying our methods in practice, namely that of processing speed. Our algorithms, when run on a conventional serial computer (such as a Symbolics 3600 or a Sun 3 or 4 series) can take several minutes or even hours to complete. We believe that this long execution time and its related computational complexity are inherent in the solution to the problems and hence we must devise ways of applying additional computing power to our algorithms. This naturally leads to the study of parallel computation.



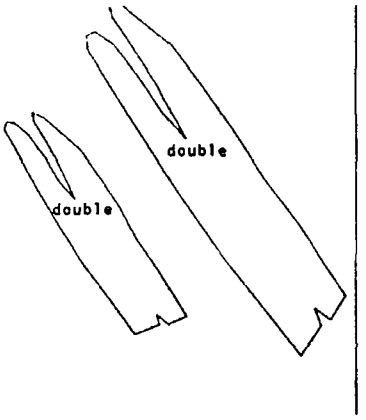
(a) Ships 160x160 image window



(b) Canny edges



(c) Single ship match



(d) Double ship match

Figure 21: Fast 2D model-based matcher applied to edges of ships

There has been significant recent activity in applying parallel processing to image understanding problems. However, much of this activity focuses on numerical computations applied to iconic data structures. While such computations are necessary and useful, they are not nearly sufficient. Our approach to image understanding is firmly based on use of symbolic representations and symbolic computations. Parallelizing such computations is significantly more complex than for iconic, numerical computations and therefore, is the focus of our parallel processing research.

Our work has included both the implementation of known computer vision algorithms on a parallel machine (The Connection Machine [17], which is a Single Instruction Multiple Data (SIMD) machine having between 16k and 64k processors), and the analysis of general techniques for implementing image understanding algorithms on parallel architectures.

Several algorithms have been implemented to evaluate the capabilities of the parallel system. The first is Adaptive Smoothing, which is an edge preserving image smoothing algorithm in which we iteratively convolve the image with a mask whose coefficients reflect

the degree of continuity of the underlying image surface [50, 49]. With a Vax front end and the parallel Lisp implementation, each iteration takes about 50 msecs on a $256 \times 256 \times 8bit$ image. In order to compare the performance to the algorithm on a serial machine, we implemented the Adaptive Smoothing on a Symbolics 3645, where it takes about 40 seconds for each iteration. Thus the speedup we get from the Connection Machine over the serial implementation is about three orders of magnitude. Using the adaptive smoothing system in a multiple scale stereo matching system based on Drumheller and Poggio [11] greatly reduces the number of possible matches at each scale and obtains a dense disparity map at fine scale.

In our work on parallel techniques for image understanding we have studied several storage and data access problems arising in mapping image algorithms onto parallel machines, parallel implementations of techniques developed by our group on hypercube and mesh based architectures, and continued our efforts in parallel computations on reconfigurable VLSI arrays and reduced meshes[1, 2]. (This work has been partially supported by AFOSR under grant AFOSR-89-0032.) We have also studied memory access systems that achieve constant time access to rows, columns, diagonals and subarrays using a minimum number of memory modules [22].

We have chosen some specific and representative medium and high level image understanding algorithms that we have found to be of general utility and are studying their mapping onto suitable parallel architectures. Our goal is not only to map these specific algorithms, but also to learn how to parallelize classes of symbolic algorithms. One specific algorithm we have focused on is a "relaxation labelling" algorithm [28]. We have found this algorithm to be useful in a variety of tasks in our work at USC; relaxation labelling has also been used by many other researchers elsewhere [48].

We have obtained several efficient parallel implementations of discrete relaxation techniques on a class of parallel architectures [24]. Using these approaches, stereo matching and other labeling problems can be solved. First, a faster sequential algorithm compared to traditional approaches for discrete relaxation is developed. This algorithm is then parallelized and mapped onto a bus-connected parallel architecture. This mapping leads to a parallel execution time of $O(nm)$ using nm processors for consistently labeling n objects with m labels. Two versions of this design are developed; one for special-purpose VLSI implementation and the other for general-purpose parallel architectures. The stereo matching technique developed in [28] can then be modified to lead to an efficient parallel implementation based on the proposed solution.

The usual approach to parallel processing is to choose a specific architecture (based on considerations of availability as well as suitability) and then attempt to map the given algorithm onto it. This often leads to complex implementations that are difficult to understand and put a severe burden on the programmer. In recent work, we are taking an alternative approach of using a flexible architecture where the architecture can be modified to suit the data flow requirements of the algorithm. Flexible architectures are becoming feasible design solutions as commercial processing elements that support parallel processing, such as the Transputer, are becoming available. Efficient parallel implementation can be achieved while maintaining the structure of the program much as it is for the serial implementation. That is, parallel efficiency can be obtained while maintaining algorithm simplicity and keeping

the programmer burden low. We have succeeded in demonstrating this approach for the relaxation algorithm; this work is described more fully in [46]. In future work, we intend to examine more complex algorithms and complete systems with this approach.

In another project, we are studying processor-time tradeoffs. These are of fundamental importance in understanding the complexity and performance of parallel computations. Driven by technological limitations, hardware cost, and flexibility, several schemes have been proposed for implementing large size computations on parallel architectures of fixed-size, or on architectures having a reduced number of processors. The major goal of such schemes is to keep the number of processors (or the processing chip-area, if implemented in VLSI) independent of the problem size and subject only to hardware cost, and other practical considerations. Such considerations are particularly important for problems on digitized images. With increasing image resolution, a processor array for a 1024×1024 image with a fixed number of pixels, say 8, per processor requires more than 10^5 processors. Design and implementation of such large arrays may be prohibitive, in addition to dealing with I/O limitations, programming and testing methodologies. Furthermore, if this array is required to handle larger size images, say images of size 2048×2048, then processor-time trade-offs must be addressed again.

Direct mapping of parallel techniques from a specific organization onto a smaller version of the same organization generally does not lead to linear processor-time trade-off. New techniques based on combining efficient parallel and sequential algorithms must be developed. We have considered several parallel architectures with a large memory and a reduced number of processors for parallel image computations [3, 4]. The memory size is proportional to the image size. However, the number of processors can be varied over a wide range while maintaining processor-time optimal performance. Architectures considered include the *reduced mesh of trees* (RMOT), *mesh-connected modules* (MCM), *linear arrays, two dimensional meshes, hypercubes*, and *shuffle-exchange* networks. An alternate cost-effective parallel architecture, designated window architecture, is proposed for image understanding applications [23]. This architecture consists of a small number of processors with mesh connections and a large external memory with simple processor-memory access scheme. Parallel solutions for several image understanding problems, such as image labeling, computing image transforms, computing geometric properties, image and stereo matching using high level primitives such as line segments, have been derived on this architecture [23].

References

[1] H. Alnuweiri and V. K. Prasanna-Kumar. Fast image labelling using local operators on mesh connected computers. In *Proceedings of the International Conference on Parallel Processing*, 1989.

[2] H. Alnuweiri and V. K. Prasanna-Kumar. Optimal image computations on reduced VLSI arrays. *IEEE Transactions on Circuits and Systems*, 1989.

[3] H. Alnuweiri and V.K. Prasanna Kumar. Optimal image algorithms on an orthogonally connected memory architecture. In *Proceedings of the International Conference on*

Pattern Recognition, pages 350–355, Atlantic City, New Jersey, June 1990.

[4] H. Alnuweiri and V.K. Prasanna Kumar. Optimal image computations on reduced processor parallel architectures. *Parallel Architectures and Algorithms for Image Understanding*, 1990. Academic Press.

[5] H. Asada and M. Brady. The curvature primal sketch. In *Proceedings of the IEEE Workshop on Computer Vision: Representation and Control*, pages 8–17, Annapolis, Maryland, May 1984.

[6] H. Blum. *A Transformation for Extracting New Descriptors of Shape.* MIT Press, Cambridge, MA, 1967.

[7] R. A. Brooks. Symbolic reasoning among 3-D models and 2-D images. *Artificial Intelligence*, 17:285–349, 1981.

[8] J.-S. Chen. *Accurate Edge Detection for Multiscale Processing.* PhD thesis, University of Southern California, 1989.

[9] S. D. Cochran and G. Medioni. Accurate surface description from binocular stereo. In *Proceedings of the DARPA Image Understanding Workshop*, Palo Alto, California, May 1989.

[10] S. D. Cochran and G. Medioni. Accurate surface description from binocular stereo. In *Proceedings of the Workshop in Interpretation of 3D Scenes*, pages 16–23, Austin, Texas, November 27–29 1989.

[11] M. Drumheller and T. Poggio. On parallel stereo. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 1439–1448, San Francisco, California, April 1986.

[12] T.-J. Fan. *Describing and Recognizing 3-D Objects Using Surface Properties.* PhD thesis, University of Southern California, August 1988. Technical Report IRIS-237.

[13] T.-J. Fan, G. Medioni, and R. Nevatia. Segmented descriptions of 3-D surfaces. *IEEE Journal of Robotics and Automation*, RA-3(6):527–538, December 1987.

[14] T.-J. Fan, G. Medioni, and R. Nevatia. Recognizing 3-d objects using surface descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(11):1140–1157, November 1989.

[15] F.P. Ferrie and M.D. Levine. Integrating information from multiple views. In *Proceedings of the IEEE Workshop on Computer Vision*, pages 117–122, December 1987.

[16] P. Fua and Y. G. Leclerc. Model driven edge detection. In *Proceedings of the DARPA Image Understanding Workshop*, volume 2, pages 1016–1021, Cambridge, Massachusetts, April 1988.

[17] D. Hillis. *The Connection Machine.* MIT Press, Cambridge, Massachusetts, 1985.

[18] W. Hoff and N. Ahuja. Surfaces from stereo: Integrating feature matching, disparity estimation, and contour detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(2):121–136, February 1989.

[19] A. Huertas, W. Cole, and R. Nevatia. Detecting runways in complex airport scenes. *Computer Vision, Graphics, and Image Processing*, August 1990.

[20] A. Huertas and R. Nevatia. Detecting buildings in aerial images. *Computer Vision, Graphics, and Image Processing*, 41(2):131–152, February 1988.

[21] J. L. Jezouin, P. Saint-Marc, and G. Medioni. Building an accurate range finder with off the shelf components. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 195–202, Ann Arbor, Michigan, June 1988.

[22] K. Kim and V. K. Prasanna-Kumar. Parallel memory systems for image processing. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, San Diego, California, June 1989.

[23] W.-M. Lin. Mapping image algorithms onto fixed size window architecture. USC Thesis in preparation, June 1990.

[24] W.-M. Lin and V.K. Prasanna Kumar. Parallel architectures for discrete relaxation algorithms. Technical report, University of Southern California, June 1990.

[25] D. Lowe. *Perceptual Organization and Visual Recognition.* Kluwer Academic Press, 1985.

[26] D. Lowe and T. Binford. Segmentation and aggregation: An approach to figure-ground phenomena. In *Proceedings of the DARPA Image Understanding Workshop*, Stanford, California, September 1982.

[27] D. Lowe and T. Binford. The perceptual organization of visual images: Segmentation as a basis for recognition. In *American Association for Artificial Intelligence*, Washington, D.C., August 1983.

[28] G. Medioni and R. Nevatia. Matching images using linear features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):675–685, November 1984.

[29] G. Medioni and R. Nevatia. Segment-based stereo matching. *Computer Graphics and Image Processing*, 31(1):2–18, July 1985.

[30] S. Menet, P. Saint-Marc, and G. Medioni. B-snakes: Implementation and application to stereo. In *Proceedings of the DARPA Image Understanding Workshop*, Pittsburgh, Pennsylvania, September 1990.

[31] R. Mohan. *Perceptual Organization for Computer Vision.* PhD thesis, University of Southern California, August 1989. IRIS Technical Report 254.

[32] R. Mohan and R. Nevatia. Perceptual grouping for the detection and description of structures in aerial images. In *Proceedings of the DARPA Image Understanding Workshop*, pages 512–526, Boston, Massachusetts, April 1988.

[33] R. Mohan and R. Nevatia. Segmentation and description based on perceptual organization. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 333–341, San Diego, California, June 1989.

[34] R. Mohan and R. Nevatia. Using perceptual organization to extract 3-d structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(11):1121–1139, November 1989.

[35] J. Mundy, A. Heller, and D. Thompson. The concept of an effective viewpoint. In *Proceedings of the DARPA Image Understanding Workshop*, pages 651–659, Cambridge, MA, 1988.

[36] R. Nevatia and K. R. Babu. Linear feature extraction and description. *Computer Graphics and Image Processing*, 13:257–269, 1980.

[37] R. Nevatia and K. Price. Locating structures in aerial images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-4(5):476–484, September 1982.

[38] R. Ohlander, K. Price, and R. Reddy. Picture segmentation by a recursive region splitting method. *Computer Graphics and Image Processing*, 8:313–333, 1978.

[39] S. I. Olsen. Stereo correspondence by surface reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(3):309–315, March 1990.

[40] B. Parvin and G. Medioni. A constraint satisfaction network for matching 3d objects. In *Proceedings of the International Conference on Neural Networks*, volume II, pages 281–286, Washington, D.C, June 1989.

[41] B. Parvin and G. Medioni. A dynamic system for object description and correspondence. Submitted to International Conference on Computer Vision, 1990.

[42] J. Ponce. Ribbons, symmetries, and skew symmetries. In *Proceedings of the DARPA Image Understanding Workshop*, pages 1074–1079, Cambridge, Massachusetts, 1988.

[43] J. Ponce and D. J. Kriegman. On recognizing and positionning curved 3-D objects from image contours. In *Proceedings of the DARPA Image Understanding Workshop*, pages 461–470, Palo Alto, CA, 1989.

[44] K. Rao. *Shape Description from Sparse and Imperfect Data.* PhD thesis, University of Southern California, December 1988. IRIS Technical Report 250.

[45] K. Rao, R. Nevatia, and G. Medioni. Issues in shape description and an approach for working with sparse data. In *Workshop on Spatial Reasoning and Multi-Sensor Fusion*, pages 168–177, Chicago, October 1987.

[46] C. Reinhart and R. Nevatia. Efficient parallel processing in high level vision. In *Proceedings of the DARPA Image Understanding Workshop*, Pittsburgh, Pennsylvania, September 1990.

[47] A. Rosenfeld. Axial representations of shape. *Computer Vision, Graphics, and Image Processing*, 2(33):156–173, 1986.

[48] A. Rosenfeld, R. A. Hummel, and S. W. Zucker. Scene labeling by relaxation operations. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-6(6):420–453, June 1976.

[49] P. Saint-Marc, J.-S. Chen, and G. Medioni. Adaptive smoothing: A general tool for early vision. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, San Diego, California, June 1989.

[50] P. Saint-Marc and G. Medioni. Adaptive smoothing for feature extraction. In *Proceedings of the DARPA Image Understanding Workshop*, pages 1100–1113, Boston, Massachusetts, April 1988. Morgan Kaufmann Publishers, Inc.

[51] P. Saint-Marc and G. Medioni. B-spline contour representation and symmetry detection. In *First European Conference on Computer Vision*, pages 604–606, Antibes, France, April 1990.

[52] K. Sato and S. Inokuchi. Range-imaging system utilizing nematic liquid crystal mask. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 657–661, June 1987.

[53] F. Stein and G. Medioni. Efficient two-dimensional object recognition. In *Proceedings of the International Conference on Pattern Recognition*, volume 1, pages 13–17, Atlantic City, New Jersey, June 1990.

[54] F. Stein and G. Medioni. Toss - a system for efficient three dimensional object recognition. In *Proceedings of the DARPA Image Understanding Workshop*, Pittsburgh, Pennsylvania, September 1990.

[55] K. Stevens and A. Brookes. Detecting structures by symbolic constructions on tokens. *Computer Vision, Graphics, and Image Processing*, 37(238-260), 1987.

[56] F. Ulupınar and R. Nevatia. Using symmetries for analysis of shape from contour. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 414–427, Tampa, Florida, December 1988.

[57] F. Ulupınar and R. Nevatia. Inferring shape from contour for curved surfaces. In *Proceedings of the International Conference on Pattern Recognition*, volume 1, pages 147–154, Atlantic City, New Jersey, June 1990.

[58] F. Ulupınar and R. Nevatia. Recovering shape from contour for shgcs and cgcs. In *Proceedings of the DARPA Image Understanding Workshop*, Pittsburgh, Pennsylvania, September 1990.

[59] B.C. Vemuri and J.K. Aggarwal. 3-D model construction from multiple views using range and intensity data. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 435–437, 1986.

[60] A. Witkin and J. Tenembaum. What is perceptual organization for? In *Proceedings of the International Joint Conference on Artificial Intelligence*, Karlsruhe, W. Germany, August 1983.

[61] P. Wright and N. Ashford. *Transportation Engineering: Planning and Design.* John Wiley and Sons, 1989.

3 Detailed Technical Papers

This section contains detailed papers published in previous workshop or conference proceedings and contains the technical details of our work.

- Efficient Two-Dimensional Object Recognition, F. Stein and G. Medioni. Published in the Proceedings of the 10th International Conference on Pattern Recognition, Atlantic City, New Jersey, June 16-21, 1990, pp. 13-17.
- TOSS - A System for Efficient Three Dimensional Object Recognition, F. Stein and G. Medioni. Published in the Proceedings of the DARPA Image Understanding Workshop, Pittsburgh, Pennsylvania, September, 1990, pp. 537-543.
- Inferring Shape from Contour for Curved Surfaces, F. Ulupinar and R. Nevatia. Published in the Proceedings of the 10th International Conference on Pattern Recognition, Atlantic City, New Jersey, June 16-21, 1990, pp. 147-154.
- Recovering Shape from Contour for SHGCs and CGCs, F. Ulupinar and R. Nevatia. Published in the Proceedings of the DARPA Image Understanding Workshop, Pittsburgh, Pennsylvania, September, 1990, pp. 544-556.
- B-Spline Contour Representation and Symmetry Detection, P. Saint-Marc and G. Medioni. Published in the Proceedings of the First European Conference on Computer Vision, Antibes, France, April 1990, pp. 604-606.
- Efficient Parallel Processing in High Level Vision, C. Reinhart and R. Nevatia. Published in the Proceedings of the DARPA Image Understanding Workshop, Pittsburgh, Pennsylvania, September, 1990, pp. 829-839.

Efficient Two Dimensional Object Recognition[1]

Fridtjof Stein and Gérard Medioni

Institute for Robotics and Intelligent Systems
Powell Hall 204
University of Southern California
Los Angeles, California 90089-0273
Email: stein@iris.usc.edu

Abstract

We address the problem of recognition of multiple flat objects in a cluttered environment from an arbitrary viewpoint (weak perspective). The models are acquired automatically and initially approximated by polygons with multiple line tolerances for robustness. Groups of consecutive segments (super segments) are then Gray coded and entered into a hash table. This provides the essential mechanism for indexing and fast retrieval. Once the data base of all models is built, the recognition proceeds by segmenting the scene into a polygonal approximation; the Gray code for each super segment retrieves model hypotheses from the hash table. Hypotheses are clustered if they are mutually consistent, and represent the instance of a model. Finally, the estimate of the transformation is refined. This methodology allows us to recognize models in the presence of noise, occlusion, scale, rotation, translation and weak perspective. Unlike most of the current systems, its complexity grows as $O(kN)$ when N is the number of models, and $k \ll 1$.

1 Introduction

Object recognition involves identifying a correspondence between part of an image and a particular view of a known object. This requires matching the image against stored object models to determine if any of the models could produce a portion of the image. In the last twenty years several systems have been developed to deal with the problem of model-based object recognition in a scene. Most of these systems try to solve the matching task through tree search, searching through all promising matches. There had been numerous attempts to deal with the complexity issue which, in most of these attempts, makes the recognition slow and ineffective. Often the focus of the research is directed towards the reduced task of recognizing only one or two objects in a scene. But even then, the computational complexity is exponential for nontrivial scenes.

Grimson and Lozano Pérez [7, 8] describe a system which is able to recognize objects from sparse scene data. If there are m known objects with n_j segments each and s scene segments, there are $\sum_{j=1}^{m}(n_j)^s$ combinations of pairings between scene and model segments. The system tests these combinations using a constrained tree search. The number of combinations that need to be tested grows rapidly with object complexity. To meet the complexity challenge, Ullman and Huttenlocher [10, 11] suggested in their alignment approach the use of a minimal amount of information with highly descriptive features. They were the first to offer a solution with *polynomial* time complexity. The model is first aligned with an image using a small number of pairs of model and image features, and then the aligned model is compared directly against the image by mapping the object model into image coordinates. Another way to avoid explosion of complexity due to extensive search is a good indexing mechanism. Indexing can reduce the search space effectively. A smart approach based on some heuristics and under the assumption of fixed scale was described by Knoll and Jain [13] in their paper, which is based on what they call *feature indexed hypotheses*. They take advantage of the similarities and differences between model types to group candidate models. For each feature, a list is kept of where it occurs in each object type. When a match is found for a feature in an image, models are hypothesized for each object identity and orientation in the feature's list. Each of these hypotheses is then tested using a template match to determine which, if any, are correct. A system which is able to recognize models from a data base with up to 100 models was introduced by Kalvin, Schonberg, Schwartz and Sharir [12]. They assume fixed scale and concentrate their representation effort on the segmentation of the boundary in boundary parts, which are likely to belong to one model, and which they call *footprints*. These footprints are used to match a scene against a data base with a hashing scheme. They call this indexing mechanism *geometric hashing*. Another method based on indexing was suggested by Lambdan, Schwartz and Wolfson [14, 15]. They have developed an algorithm which deals successfully with the combinatorial explosion of possible interpretations of a model by using the scene coordinates as an index for a voting scheme. This method works fast for one model and not too many additional scene points which do not belong to the model in the scene. But if we have to deal with multiple models and cluttered scenes with a lot of extra points, or when a model is not in the scene the system suffers from ineffective search.

By reviewing the object recognition systems of the past we encounter the following problems, formulated as questions:

Generality Will the system work for *any* object, or do we have to use different methods for smooth or convex objects?

Stability Can the system recognize an object segmented differently (because of scale, noise, quantization, ...)?

Robustness Will it work on real data, with noise and quantization?

Viewpoint Can the system handle a wide range of viewpoints and therefore take into account translation, rotation, scale and perspective?

Multiple Instances Can we deal with multiple instances of a model in a scene?

Good Worst Case Performance Can the system still be fast when no model is in the scene?

Performance What are the effects of model size, scene size and number of models on the performance?

We address these issues in our system.

The paper is organized as follows. In Section 2 we talk about super segments and their representation. Section 3 focuses on the representation of models and scenes, their matching, the verification process and a discussion of the complexity of our system. In Section 4 we present some examples.

[1]This research was supported in part by DARPA contract F33615-87-C-1436 and an AT&T grant.

2 Representation

2.1 Basic Idea

Our representation of a model or a scene is based on a polygonal approximation. We are not dependent of any feature detection algorithm and we do not handle explicit distinguished points like corners or inflection points. Our opinion is that curvature is the most important feature of a general curve. It is invariant with regard to scale, rotation and translation. When we include some redundancy, we can also add the weak perspective to that list. By using a polygonal approximation we lose most of the curvature information, but we keep parts of this information in the angles of consecutive line segments (see also Lowe's SCERPO system [16]). Obviously, there is not a unique polygonal approximation for a curve (see for example Figure 1). Therefore, for the

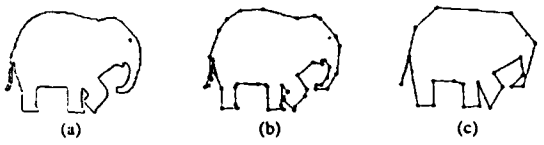


Figure 1: Elephant Shape (a) and Different Polygonal Approximations (b) and (c)

purpose of robustness, we use *several* polygonal approximations with different line fitting tolerances. Since we want to handle occlusion, we do not expect to obtain complete boundaries in our scenes, but only portions of them. On the other hand, individual segments are too local to be useful as matching primitives. Grouping a fixed number of adjacent segments provides us with our basic features, the super segments. In accordance to Figure 2,

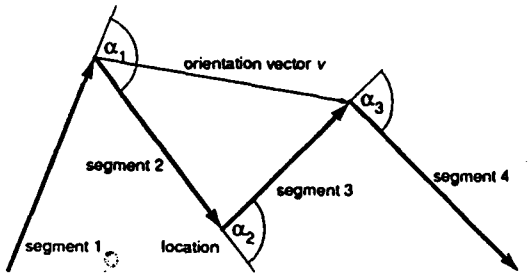


Figure 2: Super Segment

super segments are characterized by their cardinality (number of segments), angles (between consecutive segments), and the arclength (sum of the segment lengths). In addition we define the location as the middle vertex of a super segment (we only use super segments of even cardinality), the orientation (the vector between the predecessor and the successor of the middle vertex), the normal vector (the vector normal to the orientation vector), and the second moment ratio. The second moment is defined as the ellipse that can be computed from the covariance matrix of the vertices. The two eigenvectors of the covariance matrix define the axis of the ellipse. We use the eccentricity (ratio of the lengths of the two axis) as a feature (the second moment ratio) for super segments.

As mentioned before, we are mainly interested in the curvature information implicitly captured by the super segment angles. That is the reason why we use them to encode a super segment. To avoid establishing matches between super segments which have the same angles but totally different shapes, we add the measure of eccentricity (second moment ratio) to our coding scheme.

2.2 Gray Coding

One of our goals is to devise a robust representation to help reduce the problems associated with low level segmentation. Connell and Brady [3] used the Gray code [9] approach to get a difference metric for a learning system designed to learn object shapes. They developed a technique to compare different data types, and this is used in our approach. Gray coding is a generalized quantization scheme. It is not the only scheme for our method to work; other quantizations will have the same performance. However Gray coding provides a clean way of quantization and the computation of a difference without hiding the mechanisms in the system. The use of the Gray code (see Figure 3 (a)) is important in digital communication. The loss of one bit information in a Gray coded number changes the value it represents by only one. In this way, the semantic difference (difference between the values) corresponds to the syntactic difference (the Hamming distance). This property makes it useful for all applications where redundancy is desirable. In digital communication, this property means that the effect of losing any one bit is uniformly noncatastrophic. In computer vision, this property can help compare slightly different representations. First we

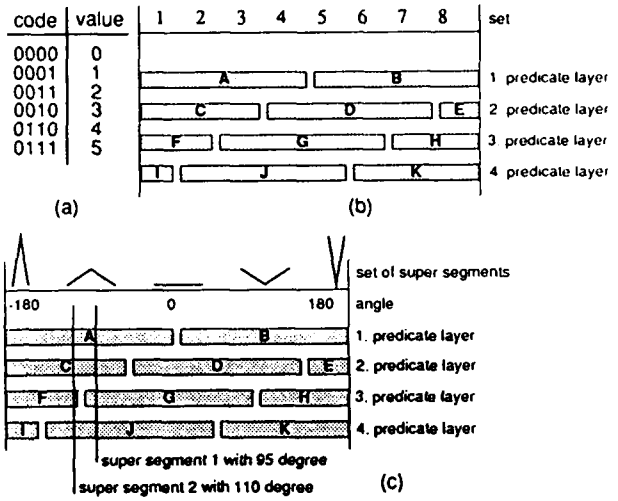


Figure 3: Gray Code (a), Gray Code for a Set (b), Gray Code for Angles (c)

have to generalize the Gray code. To encode intervals, which may either be continuous or discrete, we can use overlapping ranges. As illustrated in Figure 3 (b) we create a set of overlapping binary predicates A through K whose range cover the interval. This means that a particular value is encoded by the set of intervals it is in: 2 is encoded as $Gray(2)$ = {A, C, F, J}, 4 is encoded as $Gray(4)$ = {A, D, G, J}, and 7 is encoded as $Gray(7)$ = {B, D, H, K}. We define a difference metric as $\Delta(i,j)$ = number of different predicates between the Gray codes of i and j. Therefore we get a maximum distance Δ_{max} = number of predicate layers (in our example $\Delta_{max} = 4$). The difference between 2 and 4 is $\Delta(2,4) = 2$ and the difference between 2 and 7 is $\Delta(2,7) = 4$. The chosen Gray coding in our example is not the only possible one. By changing the range of the predicates or by adding new predicates, we can refine the resolution of our representation.

Figure 3 (c) shows the Gray coding of super segments of cardinality two. They consist of two segments and one angle between them. The angle can lie in a range $[-180°, +180°]$ and is used to Gray code the super segments. For example the Gray code of super segment 1 is encoded as {A, C, F, J}, and the Gray code of super segment 2 is encoded as {A, C, G, J}. The difference of 1 represents our intuitive feeling that the two super segments are rather similar.

Gray coding can be easily extended to higher dimensions. The

super segment in Figure 2 can be encoded by taking all the angles, encoding each one separately (with a different Gray code table for disjunct predicates) and keeping all the predicates in one set. In this example, the Gray code of the super segment is the set $\{Gray(\alpha_1), Gray(\alpha_2), Gray(\alpha_3)\} = \{[P_{11}, P_{12}, ...P_{1n}], [P_{21}, P_{22}, ...P_{2n}], [P_{31}, P_{32}, ...P_{3n}]\}$ where P_{ij} are the predicates and $n = \Delta_{max}$.

3 Recognition

3.1 Object Representation

As mentioned in the previous sections, we want to represent our model (or scene) with super segments. Therefore we first apply an edge detection algorithm on the image with the model (or scene). We use the Canny edge detector [4] for grey level images and a simple boundary tracer for binary images. The resulting edgels are further processed with a line fitting algorithm to compute the polygonal approximations. Connected linear segments form chains of adjacent segments. The segment chains provide the super segments by grouping a fixed number of adjacent segments. We then take all the super segments, encode them and take the resulting predicates as a key for a hash table, where we record the super segment as an entry (see Figure 4). That means, that every super segment is stored under the predicates, which represent the intervals, in which the angles (or other attributes) lie.

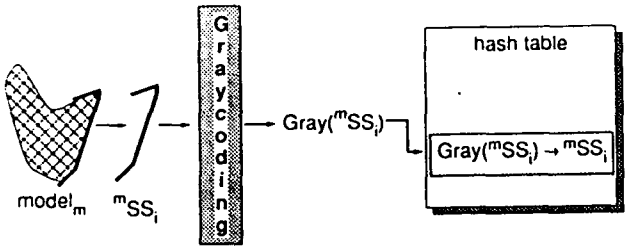


Figure 4: Representation of a Model

3.2 Matching

By using indexing for the matching process, we only select a small set of candidate models that are likely to be present in the image. We assume that most objects in our data base (hash table) are redundantly specified by their super segments. The scene is preprocessed as explained in Section 3.1 to generate all the super segments. These are Gray coded and the predicates are used to retrieve the matching hypotheses between the super segments of model and scene (see Figure 5). Two super segments s_1 and

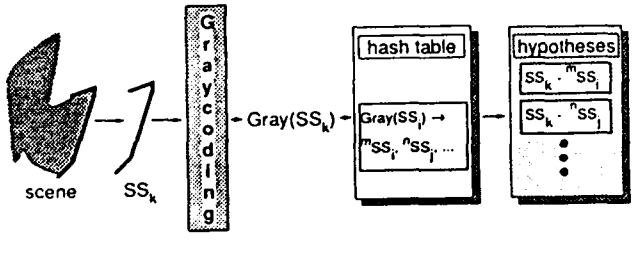


Figure 5: Matching of Model(s) with Scene

s_2 match if $\Delta(s_1, s_2) = 0$. Retrieving matches with larger Gray distance is possible by substitution of predicates with neighbor predicates.

3.3 Verification

We compute all possible matches for the super segments of the scene with the model super segments to generate multiple hypotheses. Next, we divide these n hypotheses $H = \{h_1, h_2, ...h_n\}$ according to which model the model super segment of the hypothesis belongs to. We store these into a correspondence table where we have the models m_i as keys and the i_k hypotheses $H_i = \{h^i_1, h^i_2, ...h^i_{i_k}\}$ (with $H_i \subseteq H$) as entries (see Figure 6). The

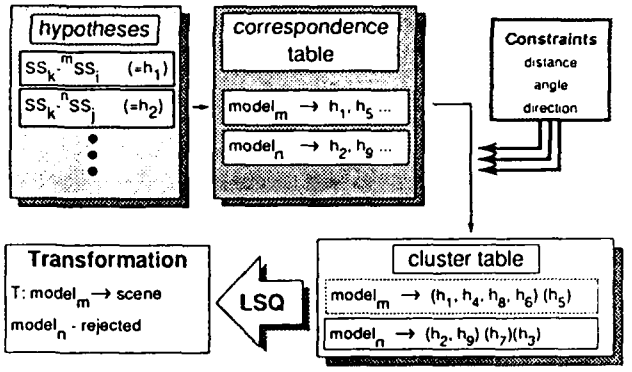


Figure 6: Verification of Hypotheses

next step is the formation of consistent clusters. For every model m_i we have to check which hypotheses h^i_x and h^i_y with $h^i_x \neq h^i_y$ are consistent with each other. We do not check every hypothesis against every other, instead, we adopt the criterion that three consistent hypotheses are sufficient to instantiate the model in the scene. If we have three consistent hypotheses $C = \{h^i_r, h^i_s, h^i_t\}$ with $C \subseteq H_i$ for one model m_i, we examine the remaining hypotheses in $H_i \setminus C$ and collect those, that are consistent with at least one of the selected three in C. When we have found one instance, represented by $I = C \cup F$, with F the additional found consistent hypotheses, we try to find more instances in the remaining hypotheses $H_i \setminus I$.

But what is meant by consistency? We use the powerful constraints (distance, angle, and direction) introduced by Grimson and Lozano-Pérez [7, 8] to prune efficiently the interpretation trees, and build our clusters. In the two dimensional domain, these three constraints define the attitude of one feature relatively to another since it specifies the three degrees of freedom (two translational and one rotational).

After we group the hypotheses into clusters which represent instances of models, we can compute the transformation from the model coordinates to the scene coordinates by applying a least squares calculation on all the matching super segments. Because of noise, we get in general a good first guess for the transformation but not an exact match. A second least squares match on corresponding corners or segments can refine the result. This is similar to the refinement procedures used by Lambdan [14, 15], Huttenlocher [10, 11], and Mundy [17].

3.4 Complexity Analysis

By addressing the complexity issue, we can study the behavior of the system when we have to deal with large data bases, meaning more than one or two models. In order to simplify the calculations, we make the assumptions that every model has the same number of super segments and that the entries are equally distributed over the hash table. When we view q (the number of super segments in the scene) as constant, we can proof that the overall cost is

$$O_{recognize} = O_{match} + O_{verify} = O(q) + O(M) = O(M),$$

where M is the number of models in the data base. It is interesting to note that the verification is the step which contributes to the linear cost, whereas the retrieval time is constant. The question remains: what is the difference between our approach and a recognition system that tries to recognize one model after the other and therefore also has the cost of $O(M)$? The main

difference lies in the slope of the linear cost function, dependent on the occurrence of a model in the scene or its absence. In our results (Section 4.4) the cost for detecting the absence of a model is less than 10% of the cost for the detection of the occurrence.

4 Examples and Performance

4.1 Parameters for Gray Coding

To encode the super segments, we need to choose the parameters for the Gray coding. What are the guidelines for setting the values for the cardinality of the super segments and the interval size of the Gray coding?

The cardinality issue reduces to the question of how local or global our representation should be. The higher the cardinality, the more global and descriptive is the representation, but we have to rely on the existence of long scene super segments belonging to one model. The lower the cardinality, the more local the description and the more *false* matches are retrieved. The answer lies in the scene:

complex shapes We should use super segments of higher cardinality (like the animal shapes in Figure 7).

simple shapes The use of super segments with lower cardinality is promising for good recognition performance (like the aircraft shapes in Figure 8).

The best answer to the question is the use of a multi cardinality representation, where we prefer matches of super segments with higher cardinality to super segments with lower cardinality. In our implementation, however, we set the cardinality to a fixed value.

The interval size of the Gray coding is the measurement of how redundant the representation is. In the scenes with the animal shapes (Figure 7) we used a relatively small interval size of 30°. For the grey level image with the aircraft scene (Figure 8) in which we have to handle noisy edges, we use a larger interval size of 60°.

4.2 Animal Shapes

We have tested our system with a library of twelve animal shapes (Figure 7 (a)). We obtained them from coarsely digitized binary images. We look for objects with a large variety of features. We want simple objects, complex objects, and objects with a lot of similarities. We set the super segment cardinality to 6. As the key for our matching we take the Gray code of the angles of the super segment and the Gray code of the ratio of the second moment of the vertices of the super segment. Each angle is encoded with an interval of 30°, which corresponds to 12 intervals. For the ratio of the second moment we take 5 intervals. Therefore the maximum matching hash table size is $5 * 12^6 \approx 15 \cdot 10^6$ entries. By putting the shapes of the twelve animals into the hash table it has a size of 1198 entries ($\approx$ 0.01 % of maximum). The animal scene was taken by printing the three animals, enlarging them, and cutting them out. Finally we put the three silhouettes on a light table and took a picture with a video camera. The camera had an angle of about 20° to the normal of the light table. This procedure guaranteed scaling, occlusion, rotation, translation, weak perspective, and noise (thanks to our high skills in cutting out - look at the ears of the giraffe). The recognition time (not including representation generation time) for the scene (see Figure 7 (b) and (c)) is 4.1 seconds on a Symbolics 3675 lisp machine.

4.3 Aircraft Shapes

In another example (see Figure 8) we try to recognize airplanes from an aerial photograph scene of an airport. The data base consists only of one airplane model (see Figure 8 (a)). We created the polygonal approximations manually. We set the super segment cardinality to 4. We take as key for our matching the Gray code of the three angles of the super segments and their second moment ratio. Each angle is encoded with an interval of 60°, which corresponds to 6 intervals. For the ratio of the second moment we take 5 intervals. Therefore the maximum matching hash table size is $5 * 6^3 = 1080$ entries. After putting the shape of the airplane into the hash table it has the size of 98 entries (= 9 % of maximum). The recognition time (not including representation generation time) for the four planes in the scene is 8.33 seconds. By looking at Table 1 it is obvious that the time to verify is relatively longer than in the animal example. The reason is that we have four instances whose hypotheses all had the same model index. To avoid complexity problems we include an intermediate indexing step. Based on the fact that super segments which belong to one airplane in the scene have similar transformation parameters, we use the translational part to preindex the hypotheses.

4.4 Large Data Base

In this example we direct our attention to a large data base. We create 200 models by randomly overlapping more than 3 and less than 7 random triangles. Our models consist, in the average, of 48 super segments. We set the super segment cardinality to 6. As the key we take the Gray code of all angles and the second moment ratio. Each angle is encoded with an interval of 60° which corresponds to 6 intervals. For the ratio of the second moment we take 5 intervals. Therefore the maximum matching hash table size is $5 * 6^5 = 38880$ entries. Our scene is generated by taking three models out of the 200. We rotate and overlap them artificially. Then we let the system recognize the scene with different data base sizes. The results are shown in the graphs in Figure 9. There are several interesting observations:

- The hash table does not grow linearly. There is a certain saturation effect. The cause is the nonequal distribution of the (encoded) super segments in the hash table.
- The three models in the scene were the first which we put into the data base. Therefore the recognition time increases very fast for the first three models. The slope is approximately 1.3 seconds per model.
- Adding more models to the database has little effect on the performance. The recognition time curve has in its linear range the slope of approximately 0.1 seconds per model. The saturation effect results in an increasing steepness. The cause is the increase of the average number of hypotheses per hash table entry.

5 Conclusion and Future Research

The results in the previous section illustrate the fact that indexing is a powerful tool. Combined with the redundant representation of the Gray code it can handle the uncertainty of segmentation, and is also fast. Our basic feature, super segments, provide us with enough information to decide whether a model might be in an image scene or not. We believe that this approach could have applications in other fields of Computer Vision.

At the moment we extend the Gray coding idea to three dimensional object recognition from three dimensional data. We have very promising results. Our goal is to be able to recognize complex three dimensional objects from two dimensional grey value images. The application of the developed method to real imagery must address the difficulties of occlusion, diffuse shadows and different lighting conditions.

We are also currently investigating the possibility of parallel implementation. The data structure of a hash table is suitable for parallel processing. Therefore we could perform the matching step in parallel. Combined with a constraint satisfaction module to perform the verification task, we dare to expect object recognition results in real time.

References

[1] N. Ayache and Faugeras O. D. HYPER: A New Approach for the Recognition and Positioning of Two-Dimensional Objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(1):44–54, 1986.

[2] N. Ayache and O. D. Faugeras. A New Method for the Recognition and Positioning of 2-D Objects. In *Proceedings of International Conference on Pattern Recognition*, pages 1274–1277, Montreal, 1984.

[3] M. Brady and J. H. Connell. Generating and Generalizing Models of Visual Objects. *Artificial Intelligence*, 31:159–183, 1987.

[4] J.F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 8(6):679–698, November 1986.

[5] M. Dhome, M. Richetin, J. T. Lapresté, and G. Rives. The Inverse Perspective Problem from a Single View. In *Proceedings of IEEE Computer Vision and Pattern Recognition*, pages 61–66, Ann Arbor, Michigan, June 1988.

[6] G. J. Ettinger. Large Hierarchical Object Recognition Using Libraries of Parameterised Model Sub-parts. In *Proceedings of IEEE Computer Vision and Pattern Recognition*, Ann Arbor, Michigan, June 1988.

[7] W. E. Grimson and T. Lozano-Pérez. Model-based recognition and localization from sparse range or tactile data. *International Journal of Robotics Research*, 3(3):3–35, 1984.

[8] W. E. Grimson and T. Lozano-Pérez. Localizing overlapping parts by searching the interpretation tree. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(4):469–482, 1987.

[9] R. W. Hamming. *Coding and Information Theory*. Prentice-Hall, Englewood Cliffs, NJ, 1980.

[10] D. P. Huttenlocher and S. Ullman. Object Recognition using Alignment. In *Proceedings of the DARPA Image Understanding Workshop*, pages 370–380, Los Angeles, California, February 1987.

[11] D. P. Huttenlocher and S. Ullman. Recognizing Solid Objects by Alignment. In *Proceedings of the DARPA Image Understanding Workshop*, April 1988.

[12] A. Kalvin, E. Schonberg, J. T. Schwartz, and M. Sharir. Two-Dimensional, Model-Based, Boundary Matching Using Footprints. *International Journal of Robotics Research*, 5(4):38–55, 1986.

[13] T. F. Knoll and R. C. Jain. Recognizing Partially Visible Objects Using Feature Indexed Hypotheses. *IEEE Journal of Robotics and Automation*, 2:3–13, March 1986.

[14] Y. Lamdan, J. T. Schwartz, and H. J. Wolfson. On Recognition of 3-D Objects from 2-D Images. In *Proceedings of IEEE International Conference on Robotics and Automation*, April 1988.

[15] Y. Lamdan and H. J. Wolfson. Geometric Hashing: A General and Efficient Model-Based Recognition Scheme. In *Proceedings of IEEE International Conference on Computer Vision*, pages 218–249, Tampa, Florida, December 1988.

[16] D. G. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31:355–395, 1987.

[17] J. L. Mundy, A. J. Heller, and D. W. Thompson. The Concept of an Effective Viewpoint. In *Proceedings of the DARPA Image Understanding Workshop*, pages 651–659, Cambridge, Massachusetts, April 1988.

[18] T. Poggio and the staff. MIT Progress in Understanding Images. In *Proceedings of the DARPA Image Understanding Workshop*, pages 56–74, Palo Alto, May 1989.

[19] L. W. Tucker, C. R. Feynman, and Fritzsche D. M. Object Recognition Using the Connection Machine. In *Proceedings of IEEE Computer Vision and Pattern Recognition*, pages 871–878, Ann Arbor, Michigan, June 1988.

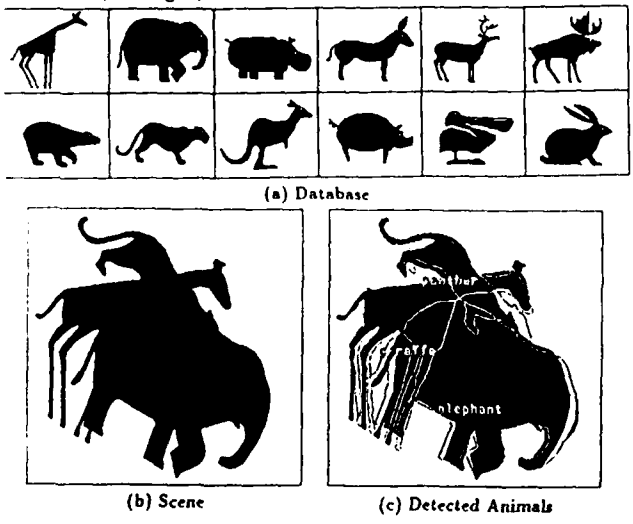


Figure 7: Animal Shapes

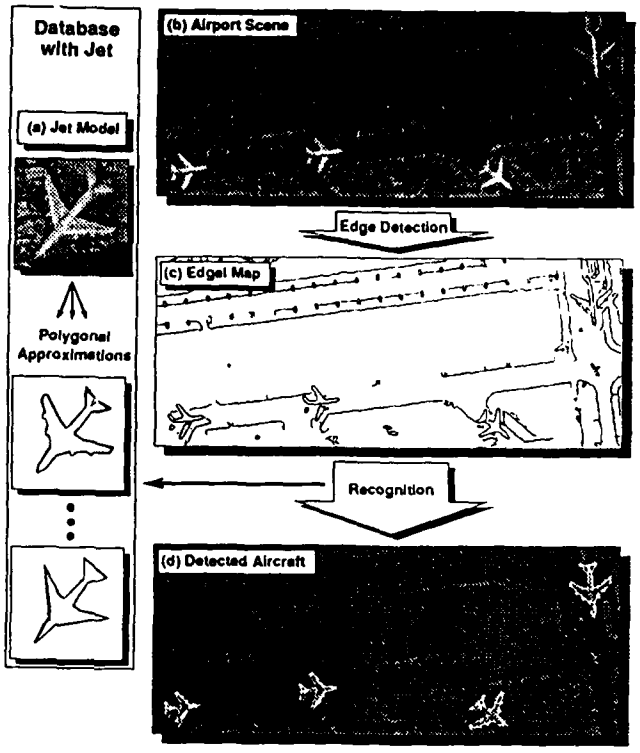


Figure 8: Aircraft Shapes

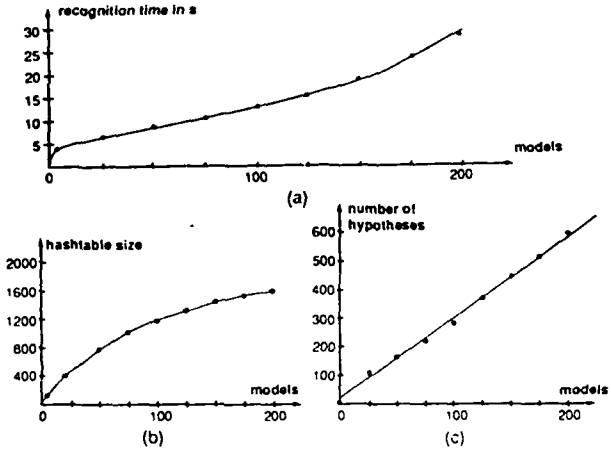


Figure 9: Recognition Time (a), Size of Hash Table (b), Number of Hypotheses (c)

scene	segments	line fit. tol.	super segments	t_{match} in sec	t_{verify} in sec
animals	101	4	86	0.64	3.46
airport	426	2	130	0.33	8.00
random	44	5	39	[0.1, 1.4]	[0.1, 31]

model	segments	line fit. tol.	super segments
giraffe	310	1.4,2,3,4,5,6,8	310
elephant	276	1.4,2,3,4,5,6,8	259
panther	298	1.4,2,3,4,5,6,8	282
jet	224	manual input	222
random	≈ 48	,5,7,9	≈ 48

Table 1: Table of Scenes and Models

TOSS - A System for Efficient Three Dimensional Object Recognition*

Fridtjof Stein and Gérard Medioni
Institute for Robotics and Intelligent Systems
Powell Hall 204
University of Southern California
Los Angeles, California 90089-0273
Email: stein@iris.usc.edu

Abstract

We present an approach for the recognition of multiple three dimensional object models from three dimensional scene data. We are addressing the problem in a realistic environment: the viewpoint is arbitrary, the objects vary widely in complexity, and we make no assumptions about the structure of the surface. We come up with a data structure which we call a *splash*. A splash consists of circular groupings of surface normals. Such a splash captures structural surface properties in a way that we can represent them by sets of two dimensional structures called super segments. Encoded super segments provide the mechanism for fast matching. The acquisition of the three dimensional models is performed automatically by computing splashes in highly structured areas of the objects. For every model all splashes are mapped on super segments. The encoded super segments are recorded in a data base. The scene is screened for highly structured areas. In these areas splashes are computed and mapped on super segments. The encoded super segments retrieve hypotheses from the data base. Clusters of mutually consistent hypotheses represent instances of models. The location of the instance in the scene is found by applying a least squares match on all corresponding points. We present results with our current system TOSS (Three dimensional Object recognition based on Super Segments) and discuss further extensions.

1 Introduction

In this paper we present an object recognition system which is able to match general three dimensional objects in an efficient way. By using the words "three dimensional" we talk about models and scenes having a three dimensional representation. By talking about "general objects" we do not make any assumptions about the shape of the objects. Matching and recognizing in an "efficient way" is based on a fast indexing and retrieval system that has a complexity which grows as $O(kN)$ when N is the number of models, and $k < 1$.

Representing a three dimensional object is either possible by using a surface or a volumetric description. Volumetric descriptions from a single view require a difficult inference step to compensate for the unseen part, so we will use descriptions based on visible surface instead. The task of object recognition involves identifying a correspondence between a part of one range image and a part of another range image with a particular view of a known object. This requires the ability to match one surface patch of one range image against a surface patch of another range image. The question is: "How can we represent a surface patch so that it can me matched in an efficient way?"

Reviewing the systems of the past, no system (known to the authors) was able to represent, match, and recognize *general* three dimensional objects. Most object recognition systems to date either rely on exact, CAD-like models, or make restrictive assumptions on the possible shape of the surface patches.

1.1 Previous Work

Grimson and Lozano Pérez [8, 9] describe a system which is able to recognize objects from sparse scene data. They exploit geometric constraints to prune the search tree of all possible matches between scene data and model data. Still, the number of combinations that need to be tested grows rapidly with object complexity. If a consistent transformation is found, the object is recognized.

Bhanu [1] presents a 3D scene analysis system for the shape matching of real world 3D objects. Object models are constructed using multiple-view range images. The object is represented as a set of planar faces approximated by polygons. Shape matching is performed by matching the face description of an unknown view with the stored model using a relaxation-based scheme called stochastic face labeling.

Horaud and Bolles [2] present the 3DPO system for recognizing and locating 3D parts in range data. The model consists of two parts: an augmented CAD model and a feature classification network. The model objects are represented by a tree-like network such that each feature contains a pointer to each instance in the CAD models. A local-feature-focus method is used for the matching process.

Faugeras and Hebert [7] developed a system to recognize and locate rigid objects in 3D space. Model objects are represented in terms of linear features such as points, lines, and planes. Range images are used as input. At first, possible pairings between model and scene features are established, the transformation is estimated using quaternions. Then, further matches are predicted and verified by the rigidity constraints.

Ikeuchi [10] developed a method for object recognition

*This research was supported in part by DARPA contract F33615-87-C-1436 and an AT&T grant.

in bin-picking tasks. The models consist of surface inertia, surface relationship, surface shape, edge relationship, extended Gaussian image, and surface characterstic distribution. Since this system is mainly designed for the task of bin-picking, only *one* type of object, which is the same one as in the model, appears in the scene.

Fan [6, 5] presents a system which takes range images as input and automatically produces a symbolic description of the objects in the scene in terms of their visible surface patches. This segmented representation may be viewed as a graph whose nodes capture information about the individual surface patches and whose links represent the relationships between them. The matching of a scene with a model is based on the comparison of the two graphs.

With 3D-POLY Chen and Kak [4] developed a system in which they present a novel approach of organizing the feature data for three dimensional objects. They present a data structure which they call *feature sphere*. The matching and verification step is based on comparing spatial relationships of special feature sets.

The closest work to our approach was done by Radack and Badler [11]. They introduce a new surface representation called *distance profile*. These profiles are used for the matching process. This method reduces the matching of three dimensional surfaces to the matching of two dimensional curves. They use points with high curvature to position the centers of the distance profiles.

Many systems were developed which based on different assumptions about shape, such as polygonal shapes, solids of revolution or generalized cylinders. In contrast, we believe that our proposed system TOSS (Three dimensional Object recognition based on Super Segments) is able to recognize rigid objects, whose shapes are not constrained by any simplifying assumptions. Our algorithm uses a representation, which is designed to capture the structure (curvature) of a surface patch and allows fast matching.

1.2 Our Previous Work

This paper describes a continuation of our early work [12] which addresses the problem of recognition of multiple flat objects in a cluttered environment from an arbitrary viewpoint (weak perspective). The models are acquired automatically and initially approximated by polygons with multiple line tolerances for robustness. Groups of consecutive linear segments (super segments) are then quantized with a Gray code and entered into a hash table. This provides the essential mechanism for indexing and fast retrieval. Once the data base of all models is built, the recognition proceeds by segmenting the scene into a polygonal approximation; the Gray code for each super segment retrieves model hypotheses from the hash table. Hypotheses are clustered if they are mutually consistent, and represent the instance of a model. Finally, the estimate of the transformation is refined. This methodology allows us to recognize models in the presence of noise, occlusion, scale, rotation, translation and weak perspective. Unlike most of the current systems, its complexity grows as $O(kN)$ when N is the number of models, and $k \ll 1$.

1.3 Plan

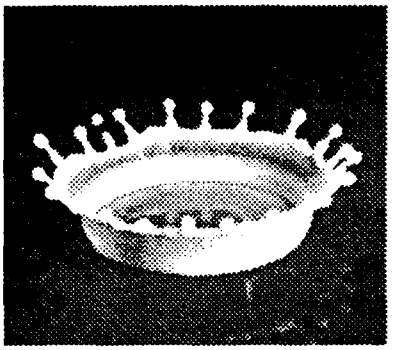
The paper is organized as follows: Section 2 introduces our basic feature, the *splash*. It consists of a group of surface normals which are mapped on a two dimensional structure. We show how we can use our two dimensional approach to represent these two dimensional structures in a way that allows us to match the three dimensional splashes. Section 3 focuses on the representation of a general three dimensional object, the matching and the verification process. In Section 4 we show results of our current implementation.

2 The Splash

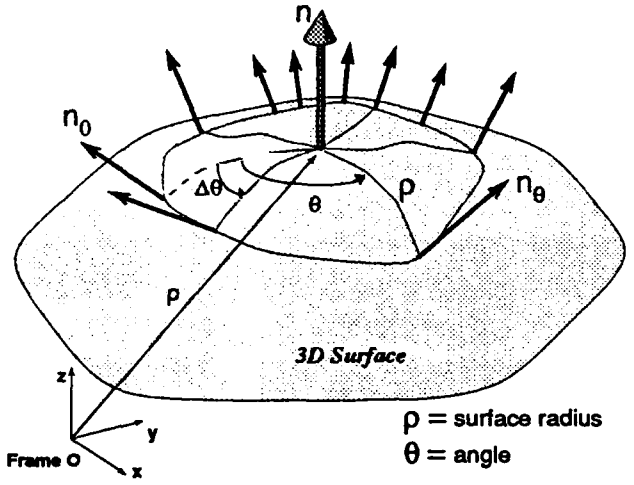
Extending the two dimensional basic feature of the super segment to three dimensions to obtain a feature which represents surfaces is awkward: the polygonal approximation of a two dimensional boundary has a property which is crucial for the super segment idea, but which is not extendable to higher dimensions: the well defined order of the neighborhood of a linear segment. Every segment on a two dimensional polygon has two adjacent neighbor segments. Based on this fact, super segments can be generated by grouping adjacent segments together. In three dimensions this ordered neighborhood property does not exist. Linear or other segmentations of a surface (or volume) lead in general to patches which can have any number and order of neighbor patches. This is a reason why we decided not to go the path of a linear (or higher order) surface segmentation to obtain a representation for matching and recognition. What are the requirements that a representation for general three dimensional objects has to meet? We want the representation to be

1. translation invariant,
2. rotation invariant,
3. general, in that we do not have to make any assumptions about the shape of the object,
4. local enough, so that we can handle occlusion,
5. robust enough, so that we can handle noise.

In the following we will use lower case to describe vectors (n, p...), and upper case to describe coordinate frames (N, O...). The basic feature for representing a general surface patch is the *splash*. The name originates from the famous picture of Professor Edgerton (MIT), showing a milk drop falling into milk (see Figure 1 (a)). This picture bears a resemblance to the normals in our basic feature. A splash is best described by Figure 1 (b). At a given location p we determine the surface normal n. We call this normal the *reference normal* of a splash. A circular slice around n with the surface radius ρ is computed. Starting at an arbitrary point on this surface circle, a surface normal is determined at every point on the circle. In practical we walk around the reference normal with a $\delta\theta$ angle (typically $1° \leq \delta\theta \leq 15°$) and obtain a set of sample points on the surface circle. The normal at the angle θ is called n_θ. A super splash is composed of splashes with different surface radii ρ_i with $i \epsilon \{1, \ldots m\}$, where m is the number of splashes in a super splash.

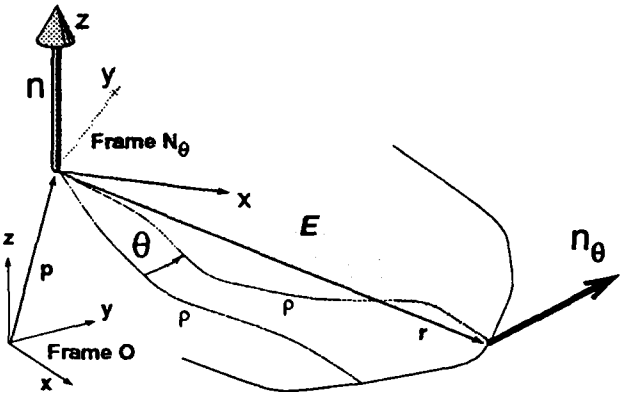


(a) Milk Splash

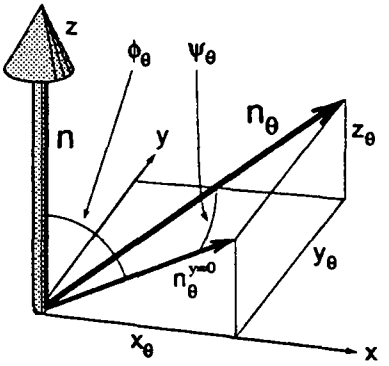


(b) Splash

Figure 1: Splashes



(a) Relationship between n and n_θ



(b) Definition of ϕ_θ and ψ_θ

Figure 2: n and n_θ

We compute a normal in our system by approximating the environment of a normal with triangles of small sizes. Every triangle votes for a triangle normal. The average of the three closest triangle normals is the surface normal. This is a very rough method, but the results were always good enough for our approach.

The frame N_θ (see Figure 2 (a)) is defined in the following way:

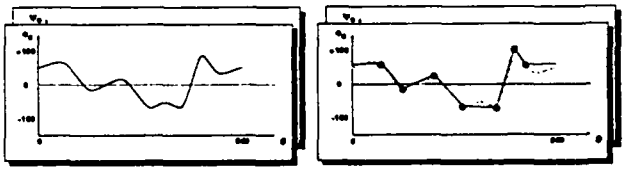
1. The surface normal n is the z axis.
2. At every location of n_θ, the location of the reference normal p and the tip of the reference normal n + p describe a plane E. The x axis is defined as the vector which is perpendicular to n and lies in the plane E. Furthermore the angle between the x axis and a vector r which is defined between the origin of Frame N_θ and the location of n_θ has to be in the interval $[-90°, 90°]$.
3. The y axis is perpendicular to the x and the z axis in a right handed coordinate system.

This frame has the property that the xy-plane always approximates the tangent plane of the surface in p. We represent n_θ in spherical coordinates: we compute the two angles ϕ_θ and ψ_θ.

$$\phi_\theta = angle(n, n_\theta^{y=0})$$

$$\psi_\theta = angle(n_\theta^{y=0}, n_\theta).$$

For every sample point of a splash we obtain such a tuple. Drawing a mapping for ϕ and ψ with respect to θ results in two mappings as in Figure 3(a). These two mappings



(a) ϕ and ψ Mapping with respect to θ

(b) Polygonal Approximation of the ϕ and ψ Mappings

Figure 3: Mapping Tuple

have the following properties:

1. Dependent on where n_0 is, the mappings are shifted along the θ axis.

2. The variation of the curve represents the structural change in the surface environment around the reference normal n.
   (a) For a splash on a sphere or plane, the mappings are constant.
   (b) A highly creased surface results in a curved set of mappings.
3. Splashes which are located close to each other have a similar shaped set of mappings. By using the word *similar* we mean similarity in the sense, so that a human would classify them as "pretty much the same". That does not automatically implicate that the pairwise difference results in small values. To be able to compare two mappings, we therefore need a metric.

At this point we have reduced the original question "How do we capture the shape of a general surface patch into a representation?" which is a three dimensional problem, into a two dimensional question "How do we capture the shape of two mappings into a representation?".

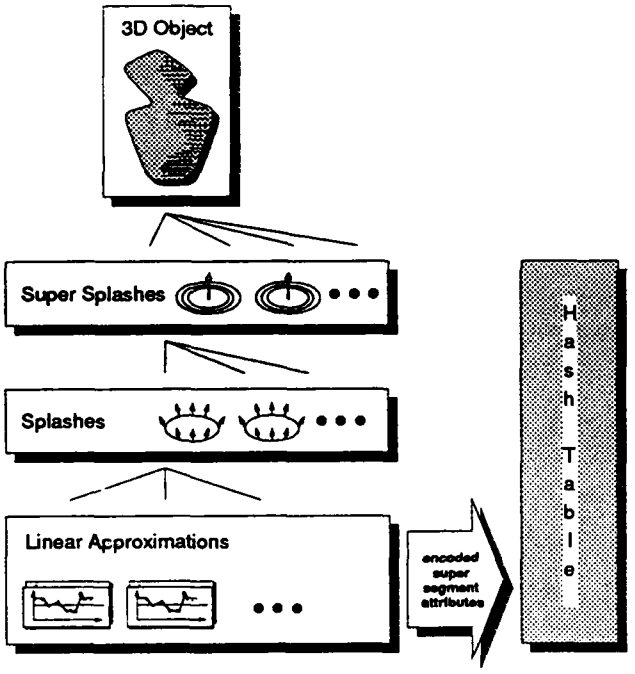


Figure 4: Representation of a Model

3 Recognition

3.1 Object Representation

The solution is straightforward based on our two dimensional approach [12].

1. For all splashes of a model we compute the mappings. In Section 3.4, we talk about the locations of the splashes.
2. For each splash the two mappings are approximated by polygonal approximations (see Figure 3(b)). It is important to note that the mappings (and therefore the polygons) are periodic ($0° = 360°$). For the purpose of robustness we use multiple line fitting tolerances. Therefore we get a set of polygons for every mapping.
3. For every polygonal approximation we compute a super segment. The start of the super segment is defined at its global maximum value. If there is more than one global maximum we use one super segment for each of the maxima. With this super segment choice, we obtain rotational invariance in our representation. By starting all super segments at the maximum of the approximation, two shifted polygons with the same shape result in the same super segment.
4. All the obtained super segments are encoded. The encoding works as described in [12]. As encodable attributes we take
   (a) the angles between two consecutive segments of a super segment (they capture the curvature information)
   (b) the mapping label ϕ or ψ
   (c) the maximum of the mapping (ϕ_{max} or ψ_{max})
   (d) the surface radius of the splash.

   Incorporated in the code of the angles of the super segments is also the cardinality (number of segments) of the super segments (by the number of angles). That avoids matching super segments of different cardinality. The encoding of ϕ_{max} or ψ_{max} allows to distinguish between different curved surfaces of the same shape (e.g. two spherical surfaces with different sphere radii). The encoding of the radius avoids matches between splashes with different splash radii.
5. All the encoded super segments serve as keys into a hash table (the data base), where we record the corresponding splashes as entries (see Figure 4).

3.2 Matching

By using indexing for the matching process, we only select a small set of candidate models that are likely to be present in the scene. We assume that most objects in our data base (hash table) are redundantly specified by their splashes. The scene is preprocessed as explained in Section 3.1 to generate all the splashes and their super segments. The encoded super segments are used to retrieve the matching hypotheses between the splashes of model and scene.

3.3 Verification

The verification stage is fully described in [12] and consists of the following steps:

1. We compute all possible matches for the splashes of the scene with the model splashes to generate multiple hypotheses.
2. These hypotheses are stored with respect to the model they vote for.

3. The next step is the formation of consistent clusters based on angle and distance constraints. A cluster of mutually consistent hypotheses represents an instance of a model.
4. After this grouping of hypotheses into clusters, we can compute the transformation from the model coordinates to the scene coordinates by applying a least squares calculation on all the matching splashes. Because of noise, we get in general a good first guess for the transformation but not an exact match. A second least squares match on corresponding corners or segments can refine the result.

3.4 Interest Operator

One question remains open: at which locations of an object should we compute the splashes? The brute force answer would be: at every pixel (in a range image). A more sophisticated answer would include the observation that we will not get *structurally rich* splashes at every point, which lead to good and unambiguous matches. Splashes in flat areas result in super segments with low cardinality. Super segments with low cardinality are less descriptive than super segments with high cardinality, which represent high structured surface patches. Therefore to obtain good and unique matches we are interested in matches of structured patches and high cardinality. These can be found at or near points of high curvature.

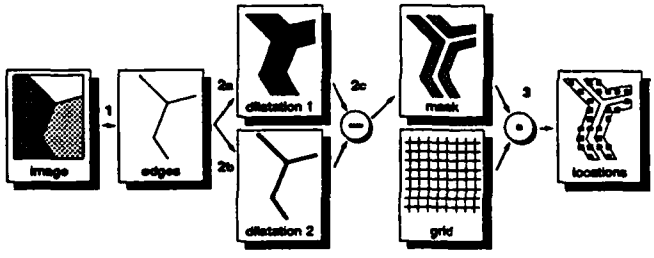


Figure 5: Interest Operator

Our simple selection method works as follows (see Figure 5):

1. We compute the edges of the artificial shaded range image (by assuming a light source at the viewer and computing a gray value for every pixel in the range image under the assumption of Lambertian conditions) with the Canny edge detector [3]. We want to position the splashes in areas where we can expect structured patches on *one* object. This property is not given on the boundary. A boundary edge typically has the object as one neighborhood and other objects or background information as the other neighborhood. Therefore we use only the "inner object edges" and throw away the boundary edges.
2. For positioning the splashes we are interested in areas around the edges. Placing a splash on a high curvature point has the disadvantage of an unreliable reference normal. A reliable reference normal is important for a stable splash. Nevertheless we want to capture the structure of the edges in the splash. Therefore the best place for a splash is in the neighborhood of an edge. We get this area in 3 steps:
   (a) We dilate the edge image by replacing every pixel on the edges by a disc of a certain radius (e.g. r_1 =8 pixels). The resulting image is called *dilatation 1*.
   (b) We dilate the edge image with another radius (e.g. r_2 =3 pixels with $r_1 > r_2$). The resulting image is called *dilatation 2*.
   (c) The subtraction of dilatation 1 and dilatation 2 gives us a mask. This mask describes an area with the above described characteristics. Points in this mask are no high curvature points, but they are close to edges.
3. We compute a grid of splashes on the range image with respect to this mask.

As we will see in Section 4, this simple method works pretty well.

3.5 Complexity Analysis

In [12] we show that for the two dimensional case, under the assumptions that every model has the same number of super segments and that the entries are equally distributed over the hash table, the overall cost is

$$O_{recognize} = O_{match} + O_{verify} = O(q) + O(M) = O(M),$$

where M is the number of models in the data base and q the number of super segments in the scene. We assume q constant to study the behavior of a large data base (large M). We show further that the slope of the linear cost function is dependent on the occurrence of a model in the scene or its absence. In our results for the two dimensional case the cost for detecting the absence of a model is less than 10% of the cost for the detection of the occurrence. In the three dimensional case, we map each splash onto a constant number of super segments. Therefore we claim that this result is also valid for the three dimensional case. To support this claim, we created a data base of 100 objects. Every object consists of a random range image. The scene is a composition of four of these objects including translation, rotation, and occlusion. In our results the cost for detecting the absence of a model is less than 50% of the cost for the detection of the occurrence. We believe that the cause for the higher relative cost of absence detection in three dimensions compared to two dimensions lies in the fact that splashes for surfaces are less descriptive than super segments for boundaries.

4 Results

With the current implementation we are able to show that the proposed recognition mechanism of recognizing general three dimensional objects works. We choose two different scenes:

1. A Mozart bust, which is highly curved, and which partially has a structured surface. Because of lack of data we cannot deal with a real three dimensional

model and a scene which consists of range data. Therefore we take the original data of the Mozart bust as model, rotate the range data synthetically to obtain the scene. We rotate pixel by pixel and fill the holes by averaging the values of neighbor pixels. This rotation process is guaranteed to add a lot of noise!

2. A scene composed of a plane and a wagon, which shows that our method works for objects which can be approximated by polygonal surfaces. We have four range images, two of the plane from different views and two of the wagon from different views. One wagon and one plane image serve as models. The scene is composed synthetically by combining the other two range images into the scene image.

4.1 Mozart

Our input data is the range image. For better visibility we show the artificially shaded images. Figure 6(a) shows the model of the Mozart bust, Figure 6(d) shows the scene, which is the model rotated by 20 degrees around a tilted axis. The inner edges are shown in Figure 6(b) and (e). The results of our interest operator are the masks shown in Figure 6(c) and (f). The recognition result is shown in Figure 7. (We overlayed a grid of the range image of the model, transformed by the resulting transformation on top of the Figure 6(d).)

4.2 Plane and Wagon

Figure 8(a) shows the shaded range image of the plane and Figure 8(b) shows the shaded range image of the wagon. Figure 9 shows the best detected solution. It is interesting to note that the plane has no highly structured areas on the wings. Therefore we get no matches on the wings. This leads to poor correspondence for the wings (we have one degree of freedom along the axis of the plane).

object (size in pixels)	# super splashes	radii (pixels)	tolerances (degrees)	grid (pixels)
mozart (264x399)	402	20,30,40	15,20,25,30	6
scene 1 (240x390)	329	20,30,40	15,20,25,30	8
plane (507x218)	60	20,30,40	15,20,25,30	8
wagon (422x178)	367	20,30,40	15,20,25,30	6
scene 2 (458x324)	293	20,30,40	15,20,25,30	8

Table 1: Table of Objects

We can give some rough numbers about the time complexity (on a serial Symbolics machine).

1. The acquisition of one super splash (consisting of typically 3 splashes with 4 line fitting tolerances) takes about 7 seconds.
2. The Mozart bust consists of about 400 super splashes, therefore it took about 45 minutes to compute all splashes.
3. The plane consists of about 60 splashes, therefore it took about 7 minutes to compute all splashes.
4. The matching process is always below 20 seconds.
5. The verification process takes less than 3 minutes.

The items 1 to 3 are processed offline to build the data base. The recognition process itself consists of item 4 and 5. All these numbers reflect neither the high parallelism which is theoretically possible nor the data redundancy with which we work at the moment. Simple improvements can significantly increase the performance. This is the goal of our future studies.

5 Conclusion and Future Research

The results with our current implementation of the TOSS system described in the last sections show that the idea of describing the surface of an object based on splashes is powerful enough to handle complex three dimensional shapes. Our future research will extend various aspects of this mechanism. Our system is designed for the recognition of surface patches. Several other pieces of information are not used to enhance the performance. One is for example the boundary of an object. There might be a possibility of including the different boundaries derived from different views of the object by including two dimensional super segment recognition in the three dimensional recognition process. We have to study this possibility. Our long term goal is to build a recognition system which is able to recognize three dimensional models in a two dimensional gray level image.

References

[1] B. Bhanu. Representation and Shape Matching of 3-D Objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(3):340–350, 1984.

[2] R. C. Bolles and P. Horaud. 3DPO: A Three-Dimensional Part Orientation System. *International Journal of Robotics Research*, 5(3):3–26, 1986.

[3] J.F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 8(6):679–698, November 1986.

[4] C. H. Chen and A. C. Kak. A Robot Vision System for Recognizing 3-D Objects in Low-Order Polynomial Time. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(6):1535–1563, 1989.

[5] T. J. Fan. *Describing and Recognizing 3-D Objects Using Surface Properties*. Springer Verlag, New York, 1990.

[6] T. J. Fan, G. Medioni, and R. Nevatia. Recognizing 3-D Objects Using Surface Descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(11):1140–1157, 1989.

[7] O. D. Faugeras and Hebert M. The Representation, Recognition, and Locating of 3-D Objects. *International Journal of Robotics Research*, 5(3):27–52, 1986.

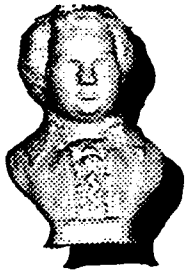
[8] W. E. L. Grimson and T. Lozano-Pérez. Model-based recognition and localization from sparse range or tactile data. *International Journal of Robotics Research*, 3(3):3–35, 1984.

[9] W. E. L. Grimson and T. Lozano-Pérez. Localizing overlapping parts by searching the interpretation tree. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(4):469–482, 1987.

[10] K. Ikeuchi. Precompiling a geometrical model into an interpretation for object recognition in bin-picking tasks. In *Proceedings of the DARPA Image Understanding Workshop*, pages 321–339, Los Angeles, California, February 1987.

[11] G. M. Radack and N. I. Badler. Local Matching of Surfaces Using a Boundary-Centered Radial Decomposition. *Journal of Computer Vision, Graphics, and Image Processing*, 45:380–396, 1989.

[12] F. Stein and G. Medioni. Graycoding and Indexing: Efficient Two Dimensional Object Recognition. In *Proceedings of International Conference on Pattern Recognition*, Atlantic City, New Jersey, June 1990.



(a) Mozart: Shaded Range Image

(b) Mozart: Inner Edges

(c) Mozart: Interest Mask

(d) Scene 1

(e) Scene 1: Inner Edges

(f) Scene 1: Interest Mask

Figure 6: Mozart Scene

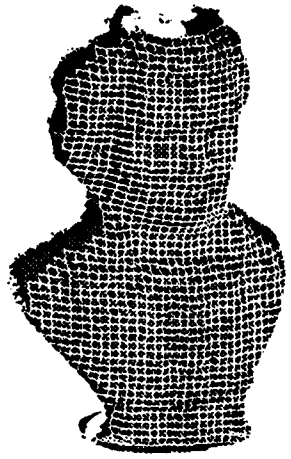
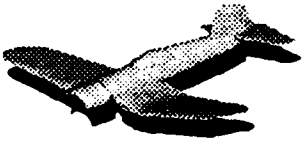
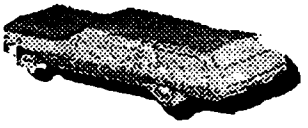


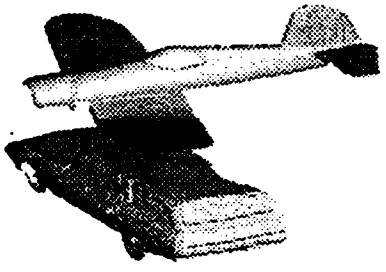
Figure 7: Scene 1: Result of Recognition



(a) Plane: Shaded Range Image



(b) Wagon: Shaded Range Image



(c) Scene 2: Shaded Range Image

Figure 8: Plane and Wagon Scene

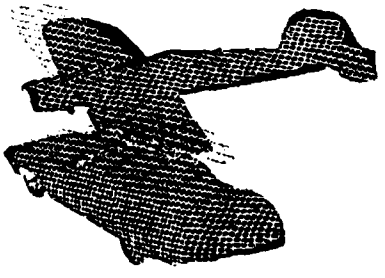


Figure 9: Scene 2: Result of Recognition

Inferring Shape from Contour for Curved Surfaces [1]

Fatih Ulupınar and Ramakant Nevatia

Institute for Robotics and Intelligent Systems
University of Southern California
Los Angeles, California 90089-0273

Abstract

We propose a technique based on analysis of symmetries in an image to infer the 3-D shape of surfaces of objects in it. This technique is analyzed and applied to Zero Gaussian Curvature surfaces in detail. Method consists of deriving a number of constraints based on a few simple assumptions. Combination of constraints to give unique (or few) solutions is discussed. Experimental results on selected scenes are given and are in good conformity with human perception. The techniques are based on concepts presented in an earlier paper and fill in important gaps in the earlier theory.

1 Introduction

Inferring shape of the surfaces in a scene from a single line drawing is an important and difficult problem in computer vision. The early work on inferring 3-D structure from a 2-D shape was focused on analysis of line drawings of polyhedra [Huf71, Clo71, Mac73, Kan81, Sug86]. There have also been some attempts at developing techniques for curved surfaces such as [BT81, Ste81, XT87, HB88]. Due to limited space, we will omit a critical analysis of previous methods. Mostly, they are not applicable to the types of scenes we analyze, except perhaps the paper by Horaud and Brady [HB88].

We propose a technique based on the analysis of symmetries in a scene. Our basic concepts were first presented in [UN88]. In that paper, we described some basic constraints that derive from observation of symmetry and other properties of line drawings. At that time, we believed that the constraints were sufficient to infer unique surface orientations for a certain class of objects. However, it turns out that even though the number of constraint equations can exceed the number of unknowns, a unique solution is not guaranteed. This paper is about how additional information can be utilized to infer unique (or a small set of) surface shapes. In the process, we have also generalized the constraints in several ways. Our method has been validated by comparison with human perception.

Throughout the paper we will assume orthographic projection, with the image plane being the $x - y$ plane. Therefore a point (x, y, z) in 3-D projects as the point (x, y) on the image plane.

In section 2, we define two types of symmetries that we find useful and the qualitative shape inferences we can draw from them. Section 3 contains our approach to

[1]This research was supported by the Defense Advanced Research Projects Agency under contract number F 33615-87-C-1436 monitored by the Air Force Wright Aeronautical Laboratories, Darpa Order No. 3119.

quantitative shape inference. We focus particularly on the analysis of objects containing "Zero-Gaussian Curvature" (ZGC) surfaces. We also give experimental results from our implementation on some selected objects.

2 Qualitative Shape Inference

We believe that symmetries have an important role in shape perception, this also has been noted and used by many researchers [NB77, Nal87, Rao88, Kan81, Ste81]. We define two types of symmetries, that we call *parallel symmetry* and *mirror symmetry*, and discuss how they can be used to infer surface shape. For curves to be symmetric (parallel or mirror) certain point-wise correspondences between two curves must exist. We will call the lines joining the corresponding points on the curves as the *lines of symmetry*, the locus of the mid points of these lines as the *axis of symmetry*, and the curves forming the symmetry as the *curves of symmetry*.

Parallel Symmetry Let $X_i(s) = (x_i(s), y_i(s))$, for $i = 1, 2$, be two curves parameterized by arc length s.
Let $\theta_i(s) = \arctan((dy_i(s)/ds)/(dx_i(s)/ds))$. Then, $X_1(s)$ and $X_2(s)$ are said to be parallel symmetric if there exists a point-wise correspondence $f(s)$ between them such that, $\theta_1(s) = \theta_2(f(s))$ for all values of s for which X_1 and X_2 are defined and $f(s)$ is a continuous monotonic function. Note that computing symmetry between two curves using this definition requires estimating the function $f(s)$ as well. A useful special case is when $f(s)$ is restricted to be a linear function.

Mirror Symmetry For mirror symmetry, the point-wise correspondence should be such that the axis of the symmetry is straight, and the lines of symmetry are at a constant angle (not necessarily orthogonal) to the axis of symmetry. This definition of the mirror symmetry is similar to that of skew symmetry. We use the term mirror symmetry in the context of curved surfaces as skew symmetry has historically been used for planar surfaces only.

We now describe some *qualitative* inferences about the shape of surfaces from their symmetries. Our inferences are based on the assumption of general viewpoint defined as:
Definition 1 General Viewpoint : *A scene is said to be imaged from a general viewpoint, if perceptual properties of the image are preserved under slight variations of the viewing direction.*

Specifically, the properties we are interested in are: straightness and parallelity of lines and symmetry of curves.

It will be useful to consider figures as belonging to one of the following three classes:

Case I: Here, *one* symmetry covers the entire boundary of the surface (though more than one such description may be possible). Figure 1 shows two examples. It can be shown that this symmetry must be a mirror symmetry. We now show that such surfaces must be planar under general viewpoint assumption.

Theorem 1 *If a surface, bounded by real edges (i.e.edges that do not change with the viewpoint), produces a line drawing in the image plane which belongs to case I, then the surface must be planar (under the assumption of general viewpoint).*

Proof: The assumption of general viewpoint implies that parallel lines in the image plane must be the projection of parallel 3-D lines, otherwise they would not project parallel from another viewpoint. Therefore we conclude that the 3-D lines, say l_i, that project as the lines of mirror symmetry on the image plane, must be parallel to each other in 3-D, because lines of mirror symmetry are parallel to each other in the image

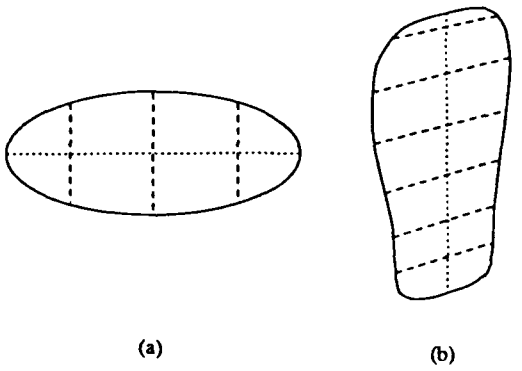


Figure 1: Two examples of case I.

plane. The axis of symmetry in 3-D, which can be obtained by joining the midpoints of the 3-D lines l_i, has to be straight because its projection on the image plane, which is the axis of mirror symmetry is straight (by definition). Therefore, the lines l_i have to lie on a plane, because they are parallel to each other and a single line, the 3-D axis of symmetry, intersects them. Hence the 3-D surface, which contains the lines l_i is planar. □

Case II: Here, the boundary of the figure is covered by exactly *two* symmetries, and at least one of which must be a parallel symmetry. We will argue that case II figures are the ones that give us the most information about the surface shape and that such cases are common in scenes of everyday experience. Figure 2 shows some examples of this case. The type of surface we perceive depends on the properties of the symmetries. We consider the case of Zero Gaussian Curvature (ZGC) surfaces first.

Lemma 1 *Parallel symmetric curves in the image plane must be projections of parallel symmetric curves in 3-D if imaged from a general viewpoint.*

Proof: This is an extension of the result that parallel lines on the image plane must be projection of parallel 3-D lines from a general viewpoint. Two curves whose tangents are parallel continuously on the image plane must be projection of two 3-D curves, whose tangents are parallel at the same points as in the image plane, from a general viewpoint.

Theorem 2 *If a surface generates one parallel symmetry and one mirror symmetry, with straight curves of mirror symmetry, on the image plane, and the straight curves of mirror symmetry are also the lines of symmetry for parallel symmetry, then the surface must be a Zero Gaussian Curvature (ZGC) surface (assuming general viewpoint and assuming that the surface does not have any variations or fluctuations that do not produce any edges in the image plane).*

Proof: Using lemma 1 we conclude that the 3-D curves producing parallel symmetry on the image plane must be parallel symmetric. Since the mirror symmetry curves are straight on the image plane, the 3-D corresponding curves must also be straight. That is the surface embeds straight lines. Also the lines that join corresponding points on the 3-D parallel symmetry curves must be on the 3-D surface (rulings of the surface) otherwise the surface would not produce the same set of symmetries from another viewpoint. Consider the normals of the points on the surface a ruling intersect the 3-D parallel symmetry curves, that is the normals N_1 and N_2 in the figure 3. Since the tangents t_1 and t_2 are the same and of course the tangent of the ruling is constant along it, the surface normals N_1 and N_2, which are the cross products of t_1 and t_2 with the tangent of the ruling, must be the same. Therefore the tangent of the surface is constant along the ruling from the

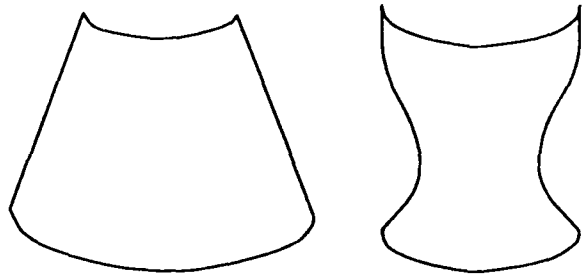


Figure 2: Examples of case II surfaces.

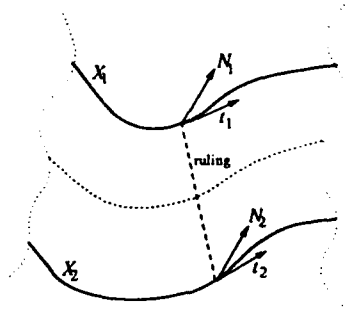


Figure 3: A ZGC surface with a ruling on it.

assumption that the surface does not have any variation that does not produce any edge. Hence the Gaussian curvature of the surface is zero. □

It follows that if the parallel symmetry has a linear correspondence function then the surface is conic, and if the correspondence function is an identity then the surface is cylindrical. A quantitative analysis for ZGC surfaces is presented in the following sections.

If the curves of mirror symmetry are not straight or have two curved parallel symmetries then we perceive a doubly curved surface (*i.e.*both of the principal curvatures are non zero). The rightmost surface in figure 2 is an example of such a doubly curved surface.

Case III: This class includes all remaining cases. We believe that in such cases, surface shape inferences can not be made directly from the given boundaries. Either no distinct shape is perceived, or shape inference assumes the existence of some boundaries that have been omitted. Figure 4 shows an example. We will not further consider such figures in this paper.

3 Quantitative Shape Recovery

We now discuss quantitative shape recovery of zero-Gaussian curvature surfaces. A Zero Gaussian Curvature (ZGC) surface is one where the the Gaussian curvature (the product of the maximum and minimum curvatures) of the surface is zero everywhere. Cylinders and cones are examples of a ZGC surface. Our analysis uses the following two theorems that apply to ZGC surfaces:

Theorem 3 *Curves obtained by intersecting a ZGC surface with two parallel planes, called the cross section plane, are parallel symmetric and the lines of symmetry are the rulings of the surface.*

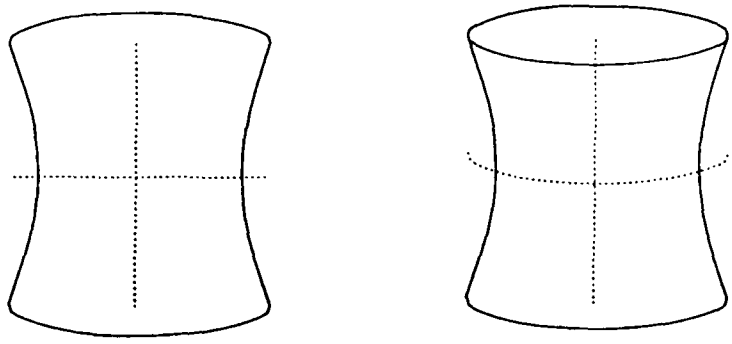


Figure 4: (a) A figure with two mirror symmetries, (b) addition of an extra curve clarifies the perceived shape

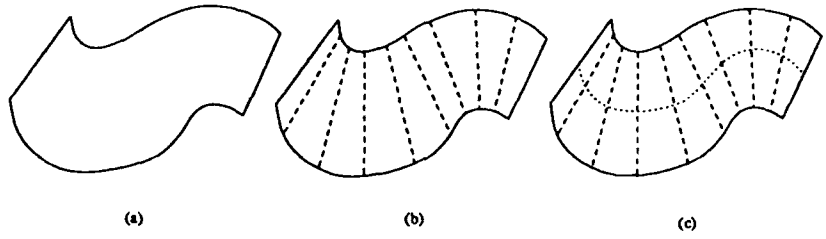


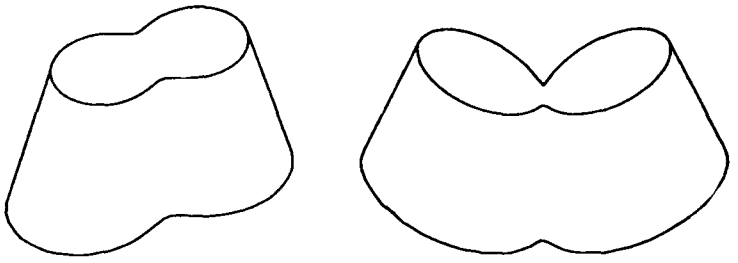
Figure 5: An example Zero Gaussian Curvature surface; (a) the boundaries only, (b) with the rulings, (c) and with axis of symmetry

Proof of this theorem is given in appendix A.1. Note that this theorem does not guarantee that parallel symmetry curves are necessarily planar. However, they do not occur by accident. For example, to obtain parallel symmetric curves from a conic surface, by cutting with non planar cross sections the cuts must be translated along the axis of the cone and scaled exactly with the scaling function of the cone. Planarity of cross sections may be confirmed by the mirror symmetry of the cross section or a segmentation into planar sections may be indicated by mirror symmetry. For example the cross section of the object in figure 6 (a) has a single mirror symmetry and is perceived planar, whereas the cross section of the object in figure 6 (b) has two mirror symmetries and the perception is that the cross section has two planar parts.

We can infer the rulings of the surface by joining the corresponding points on the two curves forming the parallel symmetry by straight lines, as shown in figure 5 (b) (the corresponding points on the two curves have the same tangent). Note that the orientation of a ZGC surface does not change along a ruling (this is also proved as a byproduct of the above proofs in the appendix). Therefore, if we find the orientation of the surface at a single point on a ruling we can extend it along the ruling. We now present the constraints for finding the surface orientations at these points.

3.1 Constraints

We now give some constraints that derive from observations of the symmetries and other boundaries in the image. We formulate three constraints discussed in sub-sections below and then discuss how to combine them.



(a) (b)

Figure 6: Objects with cross sections having (a) only one mirror symmetry, (b) two mirror symmetries

3.1.1 Curved Shared Boundary Constraint (CSBC)

This constraint relates the orientations of the two surfaces on opposite sides of an edge. The planar version has been used since early days in polyhedral scene analysis [Mac73], though the term *shared boundary constraint* is ours.

Consider two surfaces $X_1(u,v)$ and $X_2(u,v)$ meeting at a curve $\Gamma(s) = (x(s), y(s), z(s))$ as in figure 7. Since the curve $\Gamma(s)$ is generated by intersection of surfaces X_1 and X_2, $\Gamma(s)$ is a curve on both X_1 and X_2. Therefore the tangent vector $\Gamma'(s) = (x'(s), y'(s), z'(s))$ of $\Gamma(s)$ is a vector both on the tangent plane of X_1 and X_2 along the curve Γ.

Say $N_1(u,v)$ and $N_2(u,v)$ are the normals of X_1 and X_2 respectively. Along the curve $\Gamma(s)$ we can represent the normals N_1 and N_2 as $N_i(s) = N_i(u_i(s), v_i(s))$. Since $\Gamma'(s)$ is on the tangent planes of both X_1 and X_2, $\Gamma'(s)$ is orthogonal to both $N_1(s)$ and $N_2(s)$. That is

$$N_1(s) \cdot \Gamma'(s) = 0 \qquad N_2(s) \cdot \Gamma'(s) = 0 \tag{1}$$

We can rewrite it as $\Gamma'(s) \cdot (N_2(s) - N_1(s)) = 0$.

Say the normals $N_i(s)$ are represented in in $p-q$ space as $N_i(s) = (p_i(s), q_i(s), 1)$. Substituting these in the above equation gives:

$$\begin{aligned} (x'(s), y'(s), z'(s)) \cdot ((p_2(s), q_2(s), 1) - (p_1(s), q_1(s), 1)) &= 0 \\ x'(s)(p_2(s) - p_1(s)) + y'(s)(q_2(s) - q_1(s)) &= 0 \end{aligned} \tag{2}$$

This is the Curved Shared Boundary Constraint (CSBC) which states that along the curve $\Gamma(s)$ the orientation of the surfaces X_1 and X_2 are constrained by the tangent, $(x'(s), y'(s))$ of the image of the curve $\Gamma(s)$ under orthographic projection. This constraint has also been derived previously by Shafer *et al* [STK83].

A stronger constraint can be obtained if we can assume that the intersection curve, Γ, is planar. Say, Γ lies in a plane with orientation (p_c, q_c). With the assumption of planarity the constraint equation becomes:

$$x'(s)(p_c - p(s)) + y'(s)(q_c - q(s)) = 0 \tag{3}$$

We will apply this constraint to one of the curves producing parallel symmetry for a ZGC surface.

3.1.2 Inner Surface Constraint (ISC)

The inner surface constraint restricts the relative orientations of the neighboring points, within a surface. For ZGC surfaces the image of the rulings of the surface are used to constrain the surface orientation of neighboring points.

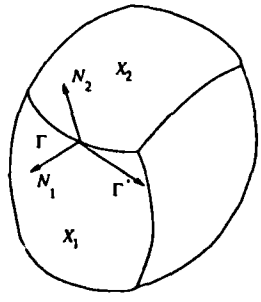


Figure 7: Two curved surfaces meeting at a curve Γ

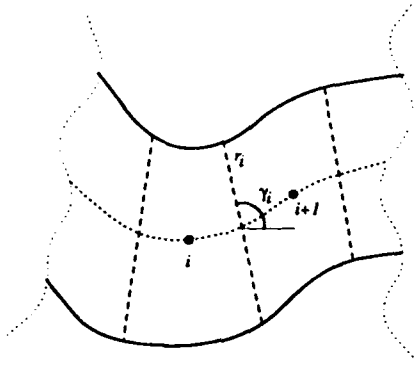


Figure 8: The inner surface constraint.

Let $X(u,v) = (x(u,v), y(u,v), z(u,v))$ be a (u,v) parametric representation of the surface X, and let v be along the direction of minimum curvature (rulings for ZGC surfaces). We can form an orientation function in terms of the parameters u and v; $O(u,v) = (p(u,v), q(u,v))$. Inner Surface Constraint (ISC) states that for a constant value of the parameter v, say v_0, as the parameter u changes the direction of the function O in the $p-q$ plane, $O_u = (p_u, q_u)$, should be orthogonal to the direction of the image of the tangent of the rulings, that is the lines of symmetry (x_v, y_v), (under orthographic projection). That is;

$$\begin{aligned} (p_u, q_u) \cdot (x_v, y_v) &= 0 \\ p_u x_v + q_u y_v &= 0 \\ \boxed{\frac{q_u}{p_u} = -\frac{x_v}{y_v}} & \end{aligned} \tag{4}$$

The proof of this property is given in appendix A.2. Geometrically ISC can be described as follows: as we move along the axis of parallel symmetry (the u parameter curve) the surface orientation should move in the $p-q$ plane in a direction orthogonal to the image of the rulings (the lines of parallel symmetry). For cylindrical surfaces, for example, this ISC curve is a straight line, since all rulings are parallel to each other. Note that this constraint does not require any regularity assumptions about the contour.

The above equation expresses the inner surface constraint in a continuous domain. In digital domain, suppose the surface orientation is to be computed at n points for a ZGC

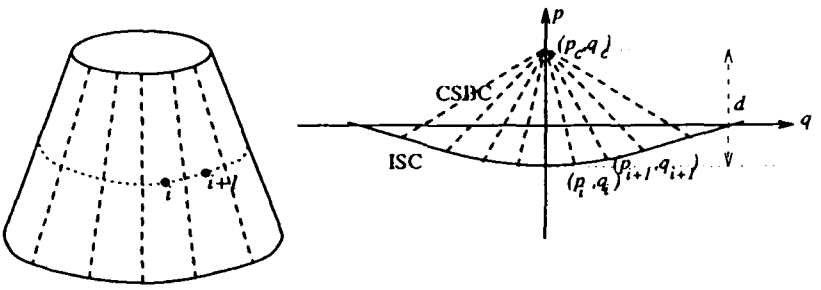


Figure 9: The three degrees of freedom present, p_c, q_c, d, in a ZGC surface after applying the constraints ISC and CSBC.

surface (these n points are along the axis of the parallel symmetry, since the surface orientation for a ZGC does not change along the rulings). We have $2n$ unknowns, (p_i, q_i) for n point. This constraint provides us with $n-1$ constraint equations as shown below.

Let the image of the ruling r_i between the i^{th} and $i+1^{st}$ points make an angle γ_i with the horizontal as in figure 8. The constraint equation relates the change in orientation along the axis of symmetry, (p_u, q_u), to the tangent of the ruling, (x_v, y_v). Here the tangent of the ruling is $(x_v, y_v) = (\cos(\gamma_i), \sin(\gamma_i))$ and the derivatives (p_u, q_u) can be approximated by first order difference as $(p_u, q_u) = (p_{i+1} - p_i, q_{i+1} - q_i)$. Substituting these in equation 4 gives

$$(p_{i+1} - p_i)\cos(\gamma_i) + (q_{i+1} - q_i)\sin(\gamma_i) = 0 \tag{5}$$

3.1.3 Combination of ISC and CSBC

In digital domain we need to quantize $(p(s), q(s))$ as (p_i, q_i) and estimate $(x'(s), y'(s))$ from the image of $\Gamma(s)$, which is $(x(s), y(s))$ under orthographic projection. If the ZGC surface is to be described at n points then there are $2n+2$ unknowns, $2n$ for the surface orientations (p_i, q_i) and 2 for the cross section plane (p_c, q_c). This constraint provides us with n constraint equations. By using the curved shared boundary constraint (CSBC) in conjunction with the inner surface constraint (ISC), we get $2n-1$ equations. This leaves us with 3 degrees of freedom for describing a ZGC surface totally.

The two constraints are shown graphically in figure 9. A ZGC surface (a frustrum) is shown in (a) with rulings and the axis of the symmetry marked on the surface. The inner surface constraint (ISC) curve is shown on the $p-q$ plane. Here the section of the ISC curve from the point (p_i, q_i) to (p_{i+1}, q_{i+1}) is orthogonal to the ruling r_i. The straight lines on the $p-q$ plane are the curved shared boundary constraints (CSBC) such that at each point i the tangent of the axis of symmetry (the dotted curve on the surface) is orthogonal to the corresponding CSBC line on the $p-q$ plane. Three parameters required to fix all the orientations (p_i, q_i) are: the orientation of the plane containing the intersection curve, (p_c, q_c), and the quantity shown as d in figure 9 which we call *angle parameter*. The angle parameter can be described as distance of the ISC curve from the point (p_c, q_c) or the angle between the cross section plane and the surface analyzed at the point d is measured (for the case of this figure the angle between the cross section surface and the middle ruling for figure 9). Specifying the length of one of the CSBC lines is enough to fix the angle parameter, d.

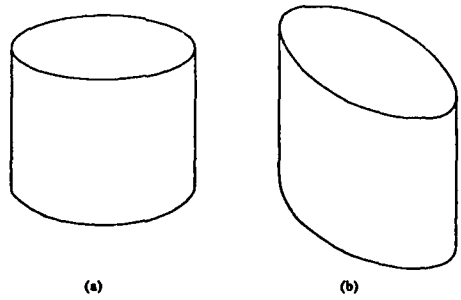


Figure 10: Two cylinders (a) is cut along the curves of maximal curvature, and (b) is cut in an arbitrary direction while preserving parallel symmetry, now we have the perception of an elliptical cylinder.

3.1.4 Orthogonality Constraint (OC)

We will assume orthogonality between the axis of parallel symmetry and the lines of parallel symmetry. This is equivalent to slicing the surface along rulings to obtain thin skew symmetric planar strips and assuming that these strips are orthogonally symmetric in 3-D, as in Kanade's analysis for polyhedra [Kan81]. This preference is illustrated in 10 where in (a) we see a circular cylinder, but in (b) we prefer to see an orthogonal elliptic cylinder rather than a slanted cylinder.

For a ZGC surface, say the tangent of the axis of symmetry makes an angle α with the horizontal and the ruling makes an angle β at some point on the surface, as in figure 11. Let the normal of the surface be $N = (p, q, 1)$ at that point. Since the 3-D tangent vectors A and B are on the tangent plane of the surface they can be represented as:

$$\begin{aligned} A &= (\cos(\alpha), \sin(\alpha), p\cos(\alpha) + q\sin(\alpha)) \\ B &= (\cos(\beta), \sin(\beta), p\cos(\beta) + q\sin(\beta)) \end{aligned} \tag{6}$$

and from the orthogonality of the 3-D vectors A and B we get: $A \cdot B = 0$ or

$$\cos(\alpha - \beta) + (p\cos\alpha + q\sin\alpha)(p\cos\beta + q\sin\beta) = 0 \tag{7}$$

This is the equation of a hyperbola in the $p - q$ space, constraining possible orientations for the surface normal N. In digital domain we need to digitize α, β, p, and q above as $\alpha_i, \beta_i, p_i, q_i$ for each point on the axis symmetry. This constraint provides us with n equations if the surface orientation is to be computed at n points. In conjunction with the previous constraints we have $2n + 2$ unknowns and $3n - 1$ equations. Therefore we now have an overconstrained case for $n > 3$. However, not all of these equations are always independent.

3.2 Combining the Constraints

The three different constraints of the previous sections provide $3n-1$ constraint equations, for n points producing $2n+2$ unknowns (including (p_c, q_c)). This suggests that the system of equations is over constrained. In other words given a general ZGC surface contours, it may not be possible to find an interpretation for the contours such that the surface obeys all the given constraints exactly. However for special but important cases, these set of constraints are dependent and may give a unique answer or even leave an extra degree of freedom. Also, even when there are no dependencies between the constraints, there may still not be a unique solution.

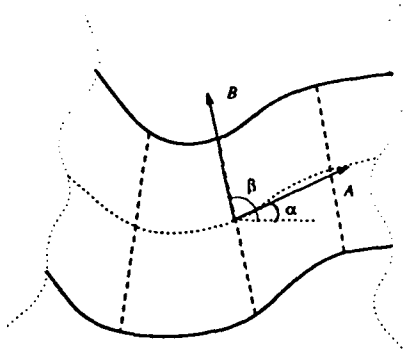


Figure 11: Orthogonality constraint

Cylindrical Surfaces: In previous work [UN88], we have shown that for a cylindrical surface, the constraints leave one degree of freedom undefined leaving the plane containing the parallel symmetry curve to be constrained to be on a line in the $p-q$ space which passes through the origin and is in the direction of the rulings.

Circular Cones: It can be shown that these constraints give a unique solution for the case of circular cone. We omit a detailed analysis for lack of space.

General ZGC Surfaces: As noted in section 3.2 for surfaces other than cylindrical surfaces and the circular cone, the three constraints can not be satisfied exactly. We believe that in most cases the planarity assumption is stronger than the orthogonality assumption. Therefore, the following process tries to maximize the orthogonality while keeping the constraints ISC and CSBC satisfied exactly.

Given a contour having parallel symmetry and a straight mirror symmetry, construct the surface orientation using constraints ISC and CSBC. With these constraints we can construct the surface orientation (p_i, q_i) at every point given the three parameters (p_c, q_c) and the angle parameter d, (see figure 9). Then choose the values (p_c, q_c) and d that minimizes the orthogonality error :

$$\Xi = \sum_{i=1}^{n} \cos\theta_i \tag{8}$$

Where θ_i is the angle between the two 3-D vectors (A and B in figure 11 whose projection on the image plane make angles α_i and β_i, with the horizontal. $\cos\theta_i$ is given by

$$\frac{(\cos(\alpha_i - \beta_i) + (p_i \cos\alpha_i + q_i \sin\alpha_i)(p_i \cos\beta_i + q_i \sin\beta_i))^2}{(1 + (p_i \cos\alpha_i + q_i \sin\alpha_i)^2)(1 + (p_i \cos\beta_i + q_i \sin\beta_i)^2)}$$

Here (p_i, q_i) are dependent on (p_c, q_c) and d as given by constraints ISC and CSBC. We want to maximize the orthogonality by minimizing the above function Ξ for (p_c, q_c) and d. We can convert this problem into a 2-D minimization problem by associating a d value to each choice of (p_c, q_c) that minimizes Ξ.

Unfortunately, for a general conic surface the global minimum for Ξ occurs when $(p_c, q_c) = (0, 0)$ and $d = \infty$; this is an infeasible interpretation. However, function Ξ, in terms of (p_c, q_c) has a "valley" of local minima (passing through the origin of the $p-q$ space) and the valley is typically a straight line. Any choice of (p_c, q_c) along this valley

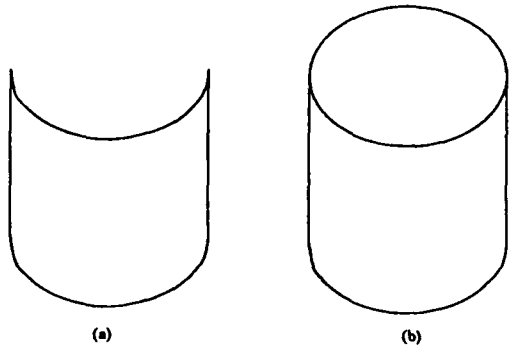


Figure 12: Two circular cylinders, (a) is without the complete cross section, (b) is with the complete cross section

is essentially equally acceptable, *i.e.*we have one degree of freedom to fix. In section 3.3 we discuss how to choose a specific value of (p_c, q_c) on that line using the shape of the cross section.

3.3 Estimating (p_c, q_c)

As discussed in section 3.2 the previous three constraints (ISC, CSBC, OC) leave one degree of freedom, namely constraining the orientation of the cross section plane, (p_c, q_c), to be along the minimum line of the function Ξ. However, computationally it is very expensive to compute the this line. Therefore we use the following gradient descent algorithm to compute (p_c, q_c).

1. Choosing a starting line, l_0, passing through the origin, in the $p-q$ space, in the direction of the mirror symmetry axis. Set the current line $l = l_0$.

2. Compute the (p_c, q_c) for the line l using the method described below.

3. Compute the value of Ξ for (p_c, q_c), check if (p_c, q_c) is along the minimum line of Ξ by repeating the above process for lines $\pm\delta\theta$ degrees off the line l, and by comparing the Ξ values for these lines.

4. If (p_c, q_c) is along the minimum line of Ξ stop. Otherwise choose another line by rotating the line l $\delta\theta$ degrees in the direction of descending Ξ. And go to step 2.

Computing (p_c, q_c) given a line l: We rotate the coordinate system such that the line l is aligned with the q axis of the $p-q$ plane then we have $p_c = 0$ and q_c is the unknown quantity.

To fix q_c, we need to use the shape of the cross-section. We propose a method for doing so that is based on perceptual properties rather than on mathematical constraints. We observe that humans prefer compact shapes but tend to avoid very high or very slow slant angles. Compactness, defined as $(area)/(perimeter)^2$ has been used previously as a compactness measure in [BY84] and [HB88]. However these methods require closed boundaries which are not always available, for example, see figure 12 (a).

Our basic method consists of fitting an ellipse to the observed contour and computing the orientation that would backproject it into the ellipse of least eccentricity, consistent with the other constraints. A correction that biases the answer towards 45° is applied.

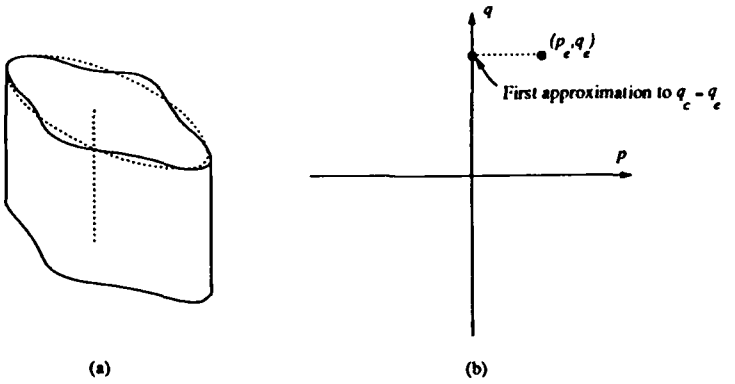


Figure 13: (a) A cylindrical object and the ellipse fitted to the cross section, (b) the orientation (p_e, q_e) that would make the ellipse a circle and its projection on the q axis gives q_e, first approximation to q_c.

First Estimation of q_c: An ellipse fitting process is utilized as a first approximation for q_c. An ellipse is fit to the cross section contour, then the orientation of the circle (p_e, q_e), that would project as the fitted ellipse is projected on the q axis, on the $p-q$ plane to obtain the first approximation of q_c, call it q_e. Figure 13 shows an example.

It may also be necessary to segment the cross section if, it is complex and repetitive. To achieve this, the concavities of the contour are found and matched. If they match in such a way that the cross section is segmented into similar pieces, then a different ellipse is fit to each piece of the contour and average of the ellipses is used to estimate q_e. Figure 14 shows various objects and ellipses fit for their cross sections.

Updating q_c: The purpose of this updating process is to simulate the bias that humans have in orienting the cross section toward 45^o. We update q_e to obtain the final q_c as follows (after converting q_c into degrees):

$$q_c = 45^o + \lambda(q_e - 45^o) \tag{9}$$

Where λ is a confidence factor in the range $[0,1]$ and is a function of how well the ellipse approximates the cross section curve. Intuition suggests that the better the approximation of the ellipse the higher the value of λ should be and the closer the q_c is to the 45^o the less the correction should be. The λ we are using :

$$\lambda(\varepsilon) = (1 - \varepsilon^2) \tag{10}$$

Where ε is the ellipse fit error (in range [0,1]). We believe that the exact form of the function is not critical. Small changes in q_c do not radically affect the perceived surface shape and humans too estimate q_c very imprecisely.

Validation : As the method described here is perceptual rather than mathematical, we performed a study on human estimation of the orientation of the cross-sections on a number of subjects and a number of test objects. Space does not allow us to describe the study in detail. In brief, we found that human perception is rather imprecise in their estimate of the desired orientation (even when measured relatively), with an average standard deviation of 8^o. We found that the results of our algorithm match the human estimates well given this large variance (the average deviation from human average was 6^o).

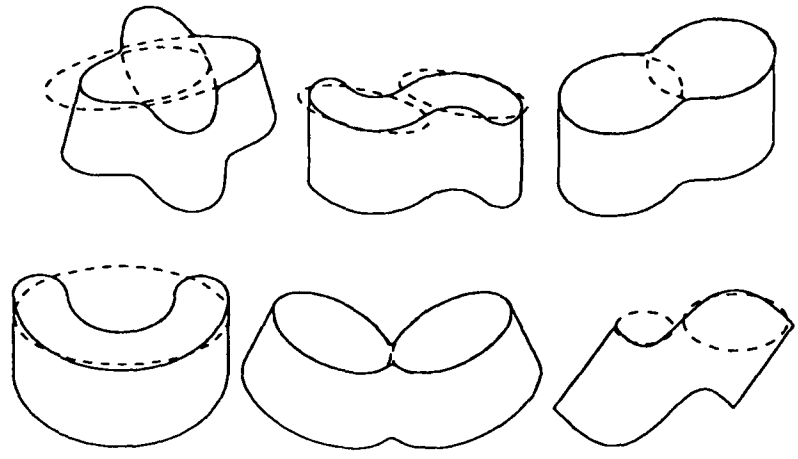


Figure 14: Objects and ellipses fit for their cross sections. For some of the objects, above, the fitted ellipses are not totally visible due to their closeness to the actual contour.

3.4 Computational Results

In figure 15 we show the computed surface orientations for various object surfaces. The symmetries of the contour curves are assumed to be given. For each object q_c is computed using the method describe in section 3.3. Then the angle parameter d is computed by minimizing the orthogonality error Ξ given in equation 8. The surface orientation, (p_i, q_i) at each point then is computed by using constraints ISC and CSBC as illustrated in figure 9. For figure 15, on the right we show the 2-D contour of the object, in the middle we show needle images of the objects computed and on the right the objects are shaded with the orientation computed for each point on the surface.

It is worth noting that for all the objects the computed orientation at the limb boundaries of the objects is orthogonal to the boundary, even though this was not an explicit constraint in our method.

The cross section of the object in the last row is segmented into two planar sections based on the observation of the mirror symmetry of the cross section. Each section is processed individually but the inner surface constraint is required to apply between the two sections of the object.

4 Conclusion

We have presented a technique for inferring shape from contour for curved surfaces. This method has been studied in depth for zero-Gaussian curvature surfaces but we believe that it extends to double curved surfaces as well; we hope to present some results on such surfaces soon. Our technique requires some assumptions about the observed contours but these assumptions are minimal, and reasonable in our view. We have also made use of observed human preferences in resolving one degree of freedom.

Our method has been implemented and tested, but it assumes that the symmetry properties are given. For real images, the symmetries are unlikely to be precise and several alternatives may be available.

The method presented only exploits the interaction between a curved surface and a

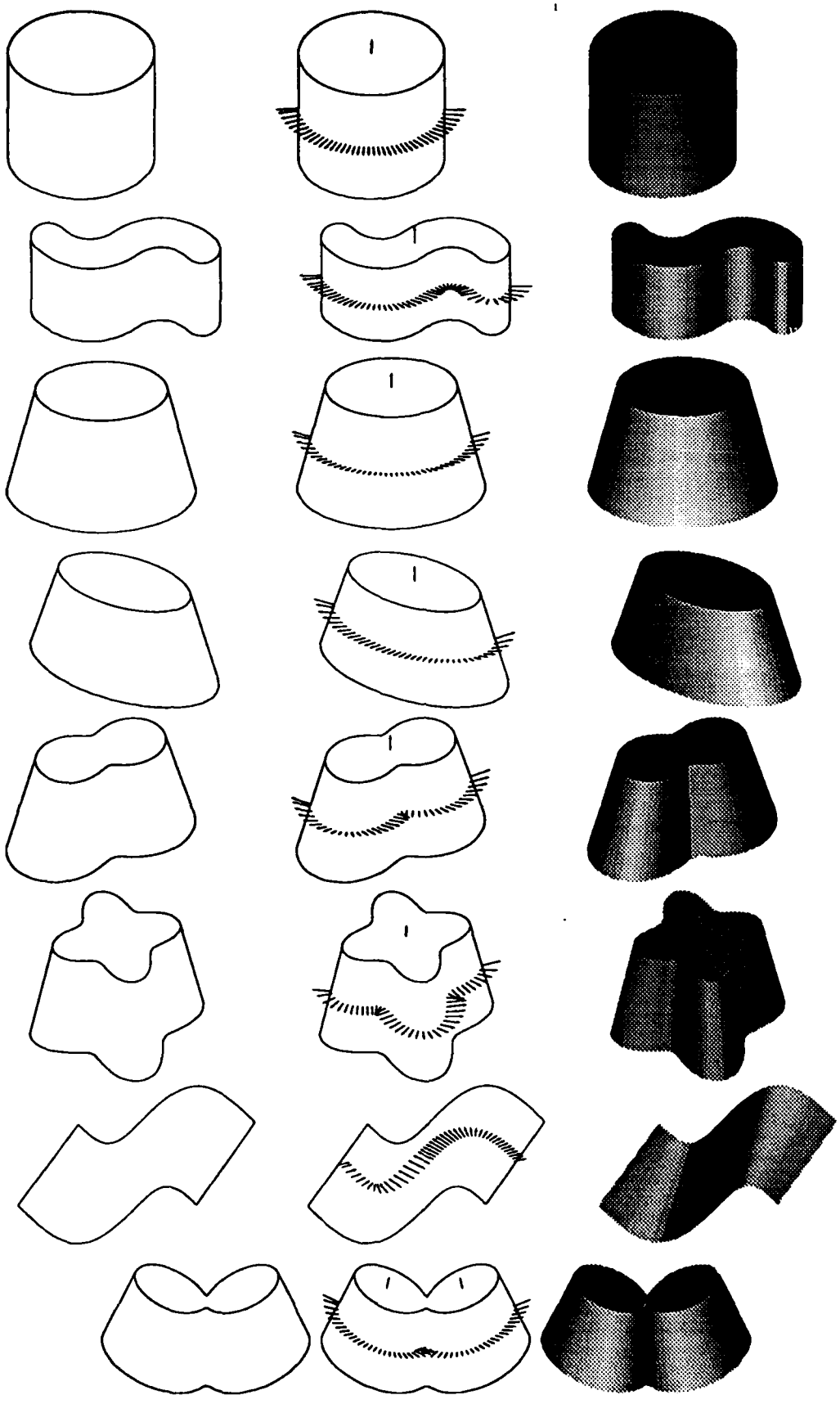


Figure 15: Sample contours, the needle images computed and their images after shading the object with the computed orientation at every point on the surface

planar surface. For more complicated objects, interactions between two (or more) curved surfaces exist. We plan to study such objects in further explorations of the described approach.

Appendix

A Proofs

In this section we give two proofs; one is related to the existence of parallel symmetries on Zero Gaussian Curvature surfaces (theorem 3) and the other proves the Inner Surface Constraint. All of the proofs uses the following surface representation.

Let $X(u,v) = (x(u,v), y(u,v), z(u,v))$ be a (u,v) parametric representation of the class C^2 Zero Gaussian Curvature surface X. Let's assume that the v parameter curves are along the lines of minimum curvature (rulings) of the surface.

Normal, $\mathcal{N}$, of this surface at any point is given by:

$$\mathcal{N} = \frac{X_u \times X_v}{|X_u \times X_v|} \tag{11}$$

where $\times$ is the vector product operator, and $|V|$ is the length of the vector V. Note here that $|\mathcal{N}| = 1$. First, I, and second, II, fundamental forms of such a surface are given by:

$$\begin{aligned} I(X_u du + X_v dv) &= E du^2 + 2F du dv + G dv^2 \\ II(X_u du + X_v dv) &= L du^2 + 2M du dv + N dv^2 \end{aligned} \tag{12}$$

where

$$\begin{aligned} E = X_u \cdot X_u \quad F = X_u \cdot X_v \quad G = X_v \cdot X_v \\ L = X_{uu} \cdot \mathcal{N} \quad M = X_{uv} \cdot \mathcal{N} \quad N = X_{vv} \cdot \mathcal{N} \end{aligned} \tag{13}$$

Since the parameter v is along the ruling (a line) the normal curvature of the surface in the direction X_v, given by $II(X_v)$, should be zero, then we have:

$$II(X_v) = N = 0 \tag{14}$$

Gaussian curvature, κ, of such a surface is given by [Lip69]

$$\kappa = \frac{LN - M^2}{EG - F^2} \tag{15}$$

Since the Gaussian curvature of the surface is zero setting $\kappa = 0$, with substituting 0 for N by equation 14 gives; $M = 0$.

A.1 Proof of Theorem 3

Consider the surface X given in section A. Also assume that the u parameter curves on the surface X are planar and parallel to each other. We have to show that the tangent of the u parameter curves, $\frac{X_u}{|X_u|}$, is constant with respect to v (*i.e.*$\frac{X_u}{|X_u|}$ is a function of u only).

Let the planes, the u parameter curves are resting on, have the normal $\mathcal{P}$ ($\mathcal{P}$ is constant). Then we have:

$$X_u \cdot \mathcal{P} = 0 \Rightarrow 0 = \frac{\partial(X_u \cdot \mathcal{P})}{\partial v} = X_{uv} \cdot \mathcal{P} + X_u \cdot \mathcal{P}_v = X_{uv} \cdot \mathcal{P} \tag{16}$$

That is $X_u \perp \mathcal{P}$ and $X_{uv} \perp \mathcal{P}$ for all u and v. Also $X_u \perp \mathcal{N}$ by equation 11 and $X_{uv} \perp \mathcal{N}$ since $M = 0$, therefore, unless $\mathcal{N} // \mathcal{P}$, we have

$$X_{uv} = c_1 \mathcal{N} \times \mathcal{P} \quad and \quad X_u = c_2 \mathcal{N} \times \mathcal{P} \tag{17}$$

for some constants c_1 and c_2. That is, $X_u // X_{uv}$, and the derivative of $\frac{X_u}{|X_u|}$ with respect to v is:

$$\frac{\partial}{\partial v}\left(\frac{X_u}{|X_u|}\right) = \frac{X_{uv}|X_u| - X_u \frac{X_u \cdot X_{uv}}{|X_u|}}{|X_u|^2} \tag{18}$$

Since $X_u // X_{uv}$ we can substitute X_{uv} by $\frac{|X_{uv}|}{|X_u|} X_u$ in the above equation:

$$\frac{\partial}{\partial v}\left(\frac{X_u}{|X_u|}\right) = X_u\left(\frac{|X_{uv}|}{|X_u|^2} - \frac{(X_u \cdot X_u)|X_{uv}|}{|X_u|^4}\right) = 0 \tag{19}$$

Therefore the tangent of the u parameter curves are parallel to each other at the points they meet a particular ruling, resulting in u parameter curves projecting as parallel symmetric with the lines of symmetry corresponding to the rulings. □

A.2 Proof of the Inner Surface Constraint

Here we will prove the inner surface constraint asserted by the equation 4.

Consider a surface as given in appendix A, a ZGC surface with v parameter curves are along the rulings and u parameter curves are arbitrary. Here we have $X_v \cdot \mathcal{N}_u = 0$ since

$$0 = \frac{\partial(X_v \cdot \mathcal{N})}{\partial u} = X_{uv} \cdot \mathcal{N} + X_v \cdot \mathcal{N}_u = M + X_v \cdot \mathcal{N}_u = X_v \cdot \mathcal{N}_u \tag{20}$$

We can write $\mathcal{N}$ in terms of the gradient (p, q) as: $\mathcal{N} = c(p, q, 1)$.

where c is the scale coefficient and equal to $(p^2 + q^2 + 1)^{-1/2}$. Differentiation of $\mathcal{N}$ with respect to the parameter u gives:

$$\begin{aligned} \mathcal{N}_u &= c_u(p, q, 1) + c(p_u, q_u, 0) \\ &= \frac{c_u}{c}\mathcal{N} + c(p_u, q_u, 0) \end{aligned} \tag{21}$$

If we set $X_v \cdot \mathcal{N}_u = 0$ from 20, where $X_v = (x_v, y_v, z_v)$ and $\mathcal{N}_u$ is given in equation 21 we get:

$$X_v \cdot \mathcal{N}_u = \frac{c_u}{c} X_v \cdot \mathcal{N} + c(x_v, y_v, z_v) \cdot (p_u, q_u, 0) = 0 \tag{22}$$

We also have $\mathcal{N} \cdot X_v = 0$ from 11. Therefore

$$\begin{aligned} x_v p_u + y_v q_u &= 0 \\ \frac{q_u}{p_u} &= \frac{x_v}{y_v} \end{aligned} \tag{23}$$

□

The same constraint can be shown to be valid for non-ZGC surfaces if the u parameter curves are chosen to be along the direction of maximum curvature.

References

[BT81] H.G. Barrow and J.M. Tenenbaum. Interpreting line drawings as three dimensional surfaces. *Artificial Intelligence*, 17:75–116, 1981.

[BY84] M. Brady and A. Yuille. An extremum principle for shape from contour. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:288–301, 1984.

[Clo71] M.B. Clowes. On seeing things. *Artificial Intelligence*, 2(1):79–116, 1971.

[HB88] R. Horaud and M. Brady. On the geometric interpretation of image contours. *Artificial Intelligence*, 37:333–353, 1988.

[Huf71] D.A. Huffman. Impossible objects as nonsense sentences. *Machine Intelligence*, 6:295–323, 1971.

[Kan81] T. Kanade. Recovery of the three-dimensional shape of an object from a single view. *Artificial Intelligence*, 17:409–460, 1981.

[Lip69] M. Lipschutz. *Differential Geometry.* McGraw-Hill, 1969.

[Mac73] A.K. Mackworth. Interpreting pictures of polyhedral scenes. *Artificial Intelligence*, 4:121–137, 1973.

[Nal87] V. Nalwa. Line drawing interpretation: Bilateral symmetry. In *Proceedings of the Image Understanding Workshop*, pages 956–967, 1987. Los Angeles.

[NB77] R. Nevatia and T.O. Binford. Description and recognition of complex-curved objects. *Artificial Intelligence*, 8:77–98, 1977.

[Rao88] Kashipati Rao. *Shape Description from Sparse and Imperfect Data.* PhD thesis, University of Southern California, 1988.

[Ste81] K. A. Stevens. The visual interpretations of surface contours. *Artificial Intelligence*, 17:47–73, 1981.

[STK83] S. A. Shafer, Kanade T., and J.R. Kender. Gradient space under orthography and perspective. *Computer Vision, Graphics and Image Processing*, 24:182–199, 1983.

[Sug86] K Sugihara. *Machine Interpretation of Line Drawings.* MIT Press, 1986.

[UN88] F. Ulupınar and R. Nevatia. Using symmetries for analysis of shape from contour. In *Proceedings of the 2nd ICCV*, pages 414–426, 1988. Florida.

[XT87] G. Xu and S. Tsuji. Inferring surfaces from boundaries. In *Proceedings of the 1st ICCV*, pages 716–720, 1987. London.

Recovering Shape from Contour for SHGCs and CGCs *

Fatih Ulupınar and Ramakant Nevatia
Institute for Robotics and Intelligent Systems
School of Engineering
University of Southern California
Los Angeles, California 90089-0273

Abstract

We analyze the properties of Straight Homogeneous Generalized Cones (SHGCs) and Constant Generalized Cylinders (CGCs), and derive the types of symmetries that the limb boundaries and cross sections of these objects produce on the image plane. The constraints on the 3-D shape of the objects are formulated based on the symmetries and from the geometry of the projection models. Finally the methods that recover the 3-D shape from the image of their contours are discussed and recovered surfaces are shown for sample objects.

1 Introduction

This paper is about inferring 3-D shape from 2-D contours for a class of objects, namely generalized cones of constant cross-section (but possibly having complex shaped axes) which we call CGCs (or *snakes*) and for straight homogeneous generalized cones or SHGCs. This class of generalized cones covers a broad class of objects of interest, it includes the so-called linear straight homogeneous generalized cones, solids of revolution, and pipes of arbitrary shape. Some examples are shown in figures 1 and 2 The method we describe is based on, and is a major generalization of, the technique we developed for inferring shape of zero-Gaussian curvature (or ZGC) surfaces [Ulupınar and Nevatia, 1988, Ulupınar and Nevatia, 1990].

Inferring shape of the surfaces in a scene from a single line drawing is an important and difficult problem in computer vision. Early work concentrated on analysis of line drawings of polyhedra [Huffman, 1971, Clowes, 1971, Mackworth, 1973, Kanade, 1981, Sugihara, 1986]. There have been other efforts at developing techniques for curved surfaces such as [Barrow and Tenenbaum, 1981, Stevens, 1981, Xu and Tsuji, 1987, Horaud and Brady, 1988]. We believe that the techniques presented here extend the complexity of surfaces that can be analyzed significantly.

*This research was supported by the Defense Advanced Research Projects Agency under contract number F 33615-87-C-1436 monitored by the Air Force Wright Aeronautical Laboratories, Darpa Order No. 3119.

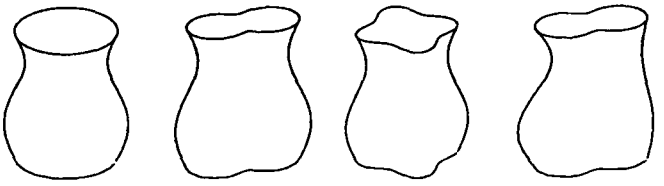


Figure 1: Sample SHGCs.

Our approach is based on an analysis of the symmetries in a scene. In section 2 we define the symmetries we use. Then we show how such symmetries arise naturally in images of the class of objects we study. In section 3 we summarize the constraints that derive from these symmetries and other properties of boundaries for determining the 3-D shape. In section 4 we give a summary of previous work on ZGC surfaces. In sections 5 and 6 we show how these constraints, and other properties of the boundary allow us to infer 3-D shape of the objects in the scene. Some computational results are also shown.

In the subsequent analysis, we will assume that the image is obtained by an orthographic projection (though some of our theorems apply to perspective projection as well) and from a general viewpoint.

Definition 1 General Viewpoint : *A scene is said to be imaged from a general viewpoint, if perceptual properties of the image are preserved under slight variations of the viewing direction.*

Specifically, the properties we are interested in are: straightness and parallelity of lines and symmetry of curves (symmetries as defined in the following).

2 Symmetry Definitions and Qualitative Shape Inference

We believe that symmetries have an important role in shape perception, this also has been noted and used by many researchers [Nevatia and Binford, 1977, Nalwa, 1987, Rao, 1988, Kanade, 1981, Stevens, 1981]. We first define two types of symmetries and then show the conditions under which they may be observed in an image of CGC or SHGC objects.

2.1 Symmetry Definitions

We define two types of symmetries, that we call *parallel symmetry* and *mirror symmetry*. For curves to be

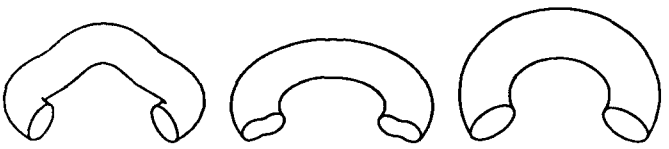


Figure 2: Some sample PRCGCs

symmetric (parallel or mirror) certain point-wise correspondences between two curves must exist. We will call the lines joining the corresponding points on the curves as the *lines of symmetry*, the locus of the mid points of these lines as the *axis of symmetry*, and the curves forming the symmetry as the *curves of symmetry*.

Parallel Symmetry Let $X_i(s) = (x_i(s), y_i(s), z_i(s))$, for $i = 1, 2$, be two curves in 3-D parameterized by arc length s.
The curves $X_1(s)$ and $X_2(s)$ are said to be parallel symmetric if there exists a point-wise correspondence $f(s)$ between them such that, $X_1'(s) = X_2'(f(s))$ for all values of s for which X_1 and X_2 are defined and $f(s)$ is a continuous monotonic function. Note that projection of curves X_1 and X_2 under orthographic projection produces image curves that are parallel symmetric such that the 3-D point correspondence is preserved. Computing symmetry between two curves using this definition requires estimating the function $f(s)$ as well. A useful special case is when $f(s)$ is restricted to be a linear function.

Mirror Symmetry For mirror symmetry, the point-wise correspondence should be such that the axis of the symmetry is straight, and the lines of symmetry are at a constant angle (not necessarily orthogonal) to the axis of symmetry. This definition of the mirror symmetry is similar to that of skew symmetry. We use the term mirror symmetry in the context of curved surfaces as skew symmetry has historically been used for planar surfaces only.

We believe that the symmetries we have defined, either separately or taken together, give some qualitative as well as quantitative information about the surface shape. In [Ulupınar and Nevatia, 1990] we showed that a figure bounded entirely by one mirror symmetry must be planar and that a figure bounded by one parallel symmetry and one mirror symmetry with straight lines of symmetry must be a ZGC surface (assuming general viewpoint in both cases). In the following we show the properties that allow us to infer the presence of PRCGCs and SHGCs.

First, we discuss some useful geometric properties of differentiable surfaces.

2.2 Surfaces and Their Limb Edges

Definition 2 *Tangent line, L_v, of a surface, S, at point, P, in a given direction, V, is the line from the point P in the direction of the tangent of the curve, C, obtained by cutting the surface by a plane, Π, that passes through P, and contains the normal, N, of the surface at P and the direction given by the vector V.*

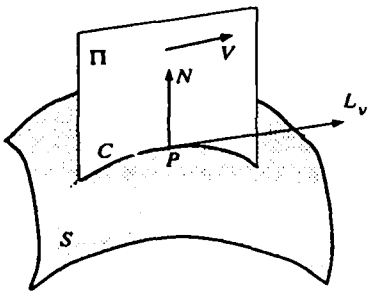


Figure 3: Tangent line, L_v, of a surface S at point P in direction V.

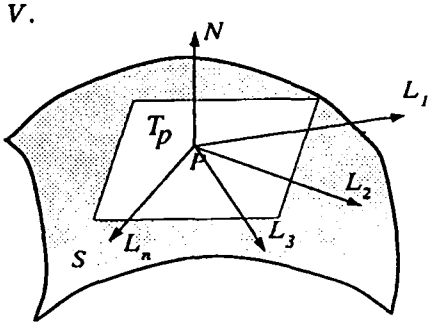


Figure 4: Tangent plane, T_p, of a surface, S, containing all the tangent lies at point P

Figure 3 shows an example.

It is a well known property in differential geometry [Do Carmo, 1976] that the tangent lines, L_{vi}, of a surface, S, at point, P, in all possible directions, $V_i \in R^3$, are on a plane, T_p, called the tangent plane of the surface at P. Moreover the plane T_p is orthogonal to the normal, N, of the surface at P. This property is shown graphically in figure 4.

Next, we define limb edges and their projections for smooth surfaces.

Definition 3 *The limb edge of a surface is a viewpoint dependent curve on the surface such that at each point on the curve the surface normal is orthogonal to the viewing direction.*

The limb edges project on the image plane as the bounding curve of the surface. At these edges the surface smoothly curves around to occlude itself. This definition of limb edges holds both for orthographic and perspective projection. Limb edges (also called "occluding contours") can give some very important information about the 3-D surface they come from; Koenderink [Koenderink, 1984] has given a nice analysis in previous work. We will show how the limb edges help us recover 3-D surface shape later in this paper.

Theorem 1 *All the tangent lines of a surface at a point, P, which is on a limb edge of the surface for a given projection geometry, project as the same line on the image plane.*

Proof The proof involves a simple combination of the definition of limb edges and the property of tangent planes. Since the normal of the tangent plane at P

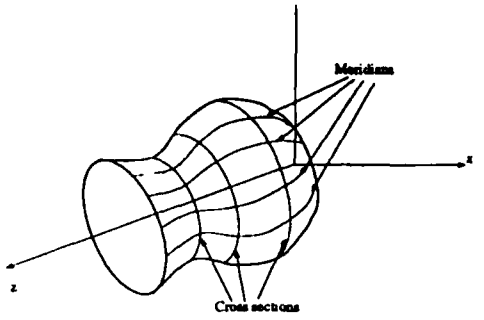


Figure 5: An SHGC along the z coordinate axis with both meridians and cross sections marked.

(which is also the normal of the surface at P) is orthogonal to the viewing direction, the tangent plane projects as a line on the image plane. Therefore all the tangent lines at P, which are included in the tangent plane also project to the one line that the plane projects into. □

This theorem, though simple and rather obvious, turns out to be highly useful in proving other important properties of limb boundaries.

2.3 SHGCs

Straight homogeneous generalized cones (SHGCs) are obtained by sliding a cross section, say C, along a straight axis, say A. The cross section is also scaled as it is swept along the axis by a scaling function, say r. We can parameterize the surface, S, of an SHGC, given the planar cross section $C(u) = (x(u), y(u), 0)$, and the scaling function $r(t)$, as :

$$S(u,t) = (r(t)x(u), r(t)y(u), t) \qquad (1)$$

The axis of the SHGC in this case is the z axis of the coordinate system. An example is shown in figure 5. Note that the cross section curves are generated by fixing t and varying u. We will call the curves generated by fixing u and varying t as the meridians of the surface. Note that cross section of an SHGC are planar because the cross section function $C(u)$ is planar, and the meridians of an SHGC are planar since the SHGC has no twist in its sweep. Let meridian edges of an SHGC be edges that are along the meridians of the SHGC. Usually images of SHGCs do not contain meridian edges, however, such edges may be present if the cross section has a tangent discontinuity (a corner). Figure 1 shows some sample SHGCs.

Theorem 2 *For an SHGC, the tangent lines of the surface in the direction of the axis from the points of any given cross section intersect at a common point on the axis of the SHGC.*

A proof of this theorem may be found in [Shafer and Kanade, 1983]. Figure 6 (a) graphically illustrates the property.

Corollary The tangents of all meridian edges at the points they intersect a single cross section intersect the axis of the SHGC at a single point. Therefore in the in the image plane, too, the tangents of the images of the meridian edges, at the point they intersect a single cross section, intersect the image of the axis in a single point, under orthographic or perspective projection.

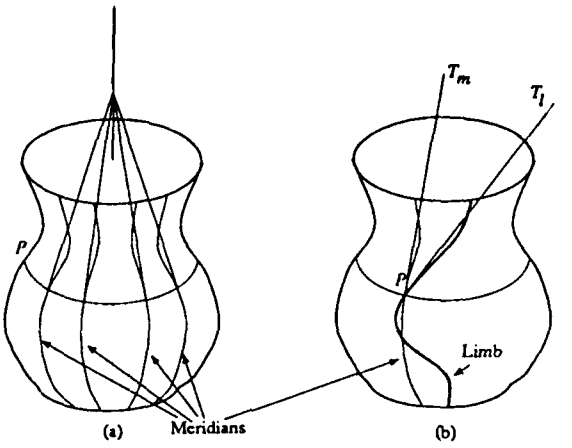


Figure 6: (a) An SHGC, and its tangent lines, in the direction of the axis emitting from a single cross section, intersecting at a single point on the axis. (b) The tangent lines, T_l, of limb edges are not the same as the tangents lines, T_m, of the meridians in 3-D.

It has been shown by Shafer[Shafer, 1983] that the limb edges on an SHGC are not planar. Therefore the limb edges of an SHGC are necessarily not along its meridians, and the tangents of the limb boundaries at the point they intersect the same cross section do not intersect the axis in 3-D. (Figure 6 (b) shows the limb edge and its tangent for an SHGC after rotating it, to show that in 3-D the tangent of the limb edge does not intersect the axis of the SHGC.) Still, it has been shown by Ponce [Ponce *et al.*, 1989] that under orthographic projection the tangents of the limb edges, at the point they intersect the same cross section, intersect the image of the axis at a single point. Here we give a simpler proof which is independent of the projection geometry.

Theorem 3 *The tangents of the projections of the limb edges at the points they intersect the same cross section, when extended, intersect the image of the axis of the SHGC at the same point.*

Proof Say the limb edge intersects a given cross section at point P (see figure 6). Since the tangent line T_m from point P in the direction of the axis of the SHGC (the tangent line of the meridian passing through the point P) intersect the axis of the SHGC, by theorem 1, the image of the tangent line T_l from point P in the direction of the tangent of the limb edge project as the same line as the tangent line T_m and thus image of the line T_l intersect the image of the axis at the same point as the image of the line T_m intersects. □

Since theorem 1 holds both under perspective and orthographic projection, the above theorem and the proof hold for both of the projection geometries.

In the following we show that the cross sections of an SHGC are parallel symmetric in 3-D with the meridian curves joining the parallel symmetric points of the cross sections.

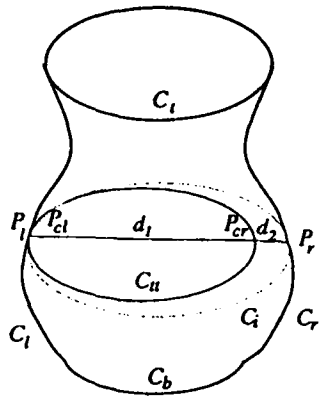


Figure 7: Image of an SHGC cut along its cross sections. Image of the top cross section curve is C_t, the bottom one is C_b and the limb boundaries are on the left C_l and on the right C_r.

Theorem 4 *The cross sections of an SHGC are parallel symmetric in 3-D with each other such that the meridian curves join the parallel symmetric points of the cross sections.*

Proof We have to show that the direction of the tangent of the cross sections is independent of the t parameter curve. Using the parameterization for an SHGC given in equation 1 the tangent of the cross sections (u parameter curves) is given by:

$$S_u = (r(t)x'(u), r(t)y'(u), 0) = r(t)(x'(u), y'(u), 0) \quad (2)$$

Clearly the direction of S_u is independent of the t parameter. □

Corollary The projection of the cross section curves of an SHGC are also parallel symmetric in the image plane. And the correspondence function is linear because cross sections are obtained by scaling a reference cross section curve without deforming it.

2.3.1 Recovering the Cross sections

We next show how to find the projections of cross sections in the image of an SHGC, given the images of its external contours. Our method does not require complete cross sections, but only the part that lies on the visible face of the SHGC. However, we require that the SHGC be cut along its cross sections, otherwise we would not have a parallel symmetry between the image curves of the two extreme cross-sections (C_t and C_b in figure 7) We conjecture that humans too do not do well if this condition is not satisfied. The following algorithm recovers the image curves C_i that correspond to the projections of the cross sections of the SHGC.

For each point $P_l \in C_l$ do:

1. Find the point $P_{cl} \in C_t$ such that $C_l'(P_l) \equiv C_t'(P_{cl})$.[1]

[1]The $\equiv$ operator is used for parallelity of vectors, that is, if $V_1 \equiv V_2$ then $V_1 = \lambda V_2$ for some scalar λ.

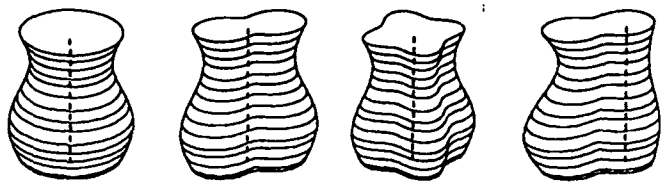


Figure 8: Images of the cross sections and axes recovered for the SHGCs in figure 1

2. Translate the cross section curve C_t such that the point $P_{cl} \in C_t$ coincides with the point P_l, obtaining the curve C_{tt}.
3. Find the point $P_{cr} \in C_{tt}$ that minimizes the function $f(P_{cr}) = (d_1 + d_2)/d_1$ which is the amount of scaling required to be applied on the curve C_{tt} to bring the point P_{cr} to the point P_r. The quantities d_1 and d_2 are the length of the line segments from P_l to P_{cr} and from P_{cr} to P_r. It can be shown that local minima of the function $f(\cdot)$ above gives the correct point $P_{cr} \in C_{tt}$ such that the limb boundary condition $C_{tt}'(P_{cr}) \equiv C_r'(P_r)$ is met.
4. Scale the curve C_{tt} by $f(P_{cr})$ so that the point P_{cr} meets with the point P_r, obtaining the curve C_i.

The curve C_i obtained by this algorithm is precisely the image of the cross section curve between the points P_l and P_r of the SHGC. Once the correspondence of the points P_l and P_r between the limb edges C_l and C_r is obtained, we can recover the image of the axis of the SHGC by using theorem 3. Figure 8 shows the computed images of the cross section curves and the axes for SHGCs in figure 1. If the parallel symmetric points of the cross section curves are joined, by theorem 4, we obtain the meridian curves.

2.3.2 Observing SHGCs

If there are two parallel symmetric curves with a linear correspondence function such that they are bound by curves that has a straight axis when the axis is computed by the above algorithm, then we can hypothesize that the line drawing results from an SHGC.

2.4 CGCs (Snakes)

Snakes are generalized cones that have a constant cross section but the axis may be an arbitrary 3-D curve. Following Shafer's terminology [Shafer *et al.*, 1983], such objects may be called CGCs. We will focus on CGCs that have planar axis and that are "right", *ie* the cross sections are orthogonal to the axis; we call such objects PRCGCs. Figure 2 shows some examples.

In the following, we show that limb boundaries of a PRCGC project as parallel symmetric curves under orthographic projection.

Let us choose a coordinate system such that the axis of the PRCGC lies in the $x - z$ plane and one of the cross-sections, say $C(u) = (c_x(u), c_y(u), 0)$, is aligned with the $x - y$ plane. Let $A(t) = (a_x(t), 0, a_z(t))$ be the axis parameterized in terms of its arc length, that is, $|A| = a_x^2 + a_z^2 = 1$ for all t. Also, let $A(0) = (0, 0, 0)$ and since the cross section is orthogonal to the axis $A'(0) =$

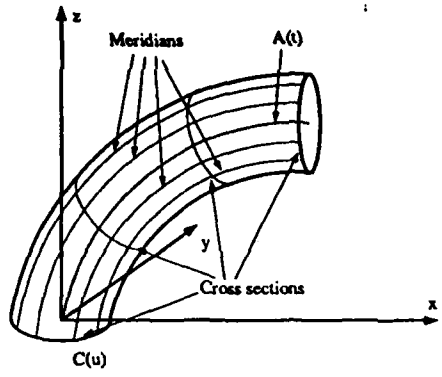


Figure 9: A PRCGC with both meridians and cross sections marked.

$(0,0,1)$. Then the surface of the PRCGC, $S(u,t)$ is given by:

$$S(u,t) = R(A'(0), A'(t)) \cdot C(u) + A(t) \qquad (3)$$

where $R(V_1, V_2)$ is the rotation matrix that transforms the direction vector V_1 into vector V_2. For $A'(0) = (0,0,1)$ and $A'(t) = (a'_x(t), 0, a'_z(t))$ the rotation matrix R becomes:

$$R = \begin{bmatrix} a'_z(t) & 0 & a'_x(t) \\ 0 & 1 & 0 \\ -a'_x(t) & 0 & a'_z(t) \end{bmatrix} \qquad (4)$$

Note that the curves generated by fixing t and varying u are the *cross sections* of the surface $S(u,t)$. We will call the curves generated by fixing u and varying t as the *meridians* of the surface. The meridians are also the loci of points on the cross section as the cross section is swept along the axis. Figure 9 shows an example.

Lemma 1 *The meridians of a PRCGC are parallel symmetric and the curves joining the parallel symmetric points of the meridians form the cross sections of the surface.*

Proof We need to show that the direction of the tangents of the surface in the direction of the meridians, $\frac{\partial S}{\partial t}$, is independent of the parameter u.

$$\begin{aligned} \frac{\partial S(u,t)}{\partial t} &= \frac{dR}{dt} \cdot C(u) + A'(t) \\ &= \begin{bmatrix} a''_z(t) & 0 & a''_x(t) \\ 0 & 0 & 0 \\ -a''_x(t) & 0 & a''_z(t) \end{bmatrix} \cdot \begin{bmatrix} c_x(u) \\ c_y(u) \\ 0 \end{bmatrix} + \begin{bmatrix} a'_x(t) \\ 0 \\ a'_z(t) \end{bmatrix} \\ &= \begin{bmatrix} a'_x(t) \\ 0 \\ a'_z(t) \end{bmatrix} + c_x(u) \begin{bmatrix} a''_z(t) \\ 0 \\ -a''_x(t) \end{bmatrix} \qquad (5) \\ &= A'(t) + c_x(u)(A''(t))^{\perp} \end{aligned}$$

where $(A''(t))^{\perp}$ is a vector which is orthogonal to the vector $A''(t)$ and is in the $x-z$ plane. Also note that, $A''(t) \cdot A'(t) = 0$ since

$$0 = d(1) = d(A'(t) \cdot A'(t)) = 2A'(t) \cdot A''(t) \qquad (6)$$

We conclude that the vector $(A''(t))^{\perp}$ is parallel to the vector $A'(t)$, since $A'(t) \perp A''(t)$, $A''(t) \perp (A''(t))^{\perp}$ and all three vectors are on a plane (the $x-z$ plane). Then, we can rewrite $\frac{\partial S}{\partial t}$ as:

$$\frac{\partial S(u,t)}{\partial t} = (1 + c_x(u)\frac{|(A''(t))^{\perp}|}{|A'(t)|})A'(t) \qquad (7)$$

It is obvious that while the length of the vector $\frac{\partial S}{\partial t}$ depends on the u parameter, the direction of it is independent of the u parameter. □

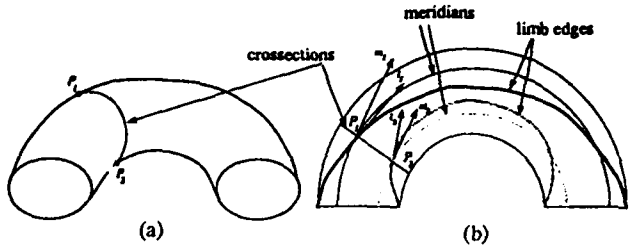


Figure 10: A PRCGC (half of a torus) (a) from a general view and (b) semi-transparent top view with the limb edges of the previous view and the meridians passing from the points P_1 and P_2 marked along with their tangent lines.

Although the meridian curves on a PRCGC are parallel symmetric it can be shown that the limb edges of a PRCGC are not necessarily parallel symmetric in 3-D (see Figure 10) However, the following theorem proves that the projections of the limb edges of a PRCGC are parallel symmetric under orthographic projection.

Theorem 5 *The limb edges of a PRCGC project as parallel symmetric curves onto the image plane.*

Proof Here we use the property given in theorem 1 and in lemma 1. Consider the points P_1 and P_2 in figure 10 such that both points are on the same cross section. As can be seen in figure 10 (b) the tangent lines l_1 and l_2 from points P_1 and P_2 in the direction of the limb edges are not parallel symmetric in 3-D. However, the tangent lines m_1 and m_2 from points P_1 and P_2 in the direction of the meridians are parallel symmetric by lemma 1. Since the tangent line l_1 project the same line as the tangent line m_1, and tangent line l_2 project the same line as the tangent line m_2 by theorem 1 the projection of the limb boundaries of a PRCGC are parallel symmetric. □

2.4.1 Observing PRCGCs

If in the image plane there are parallel symmetric curves that are terminated by two curves (possibly closed and having mirror symmetry which enhances planarity of the cross section) then we hypothesize that it is a PRCGC. The real test for the line drawing to belong to a PRCGC may be performed after the cross sections are recovered as described in section 6.1.

3 Constraints for Determining Surface Shape

We now give three constraints that derive from observations of the symmetries and other boundaries in the image.

3.1 Curved Shared Boundary Constraint (CSBC)

This constraint relates the orientations of the two surfaces on opposite sides of an edge. It is a generalization of the constraint used in polyhedral scene analysis from the early days [Mackworth, 1973] and has been stated previously in [Shafer *et al.*, 1983, Ulupınar and Nevatia, 1990].

Let two surfaces, S_1 and S_2 intersect along a curve, Γ, whose projection is the curve $\Gamma_i(s) = (\Gamma_x(s), \Gamma_y(s))$. Let the orientations of the surfaces S_1 and S_2 along the curve $\Gamma(s)$, in gradient space, be given by $(p_1(s), q_1(s))$ and $(p_2(s), q_2(s))$. Then CSBC states that:

$$\Gamma'_x(s)(p_2(s) - p_1(s)) + \Gamma'_y(s)(q_2(s) - q_1(s)) = 0 \quad (8)$$

A stronger constraint can be obtained if we can assume that the 3-D intersection curve, Γ, is planar. Say, Γ lies in a plane with orientation (p_c, q_c). With the assumption of planarity the constraint equation becomes:

$$\Gamma'_x(s)(p_c - p_i(s)) + \Gamma'_y(s)(q_c - q_i(s)) = 0 \quad ,i = 1,2 \quad (9)$$

3.2 Inner Surface Constraint (ISC)

The inner surface constraint restricts the relative orientations of the neighboring points, within a surface. Consider a curve $C(t) = (x(t), y(t), z(t))$ on a C^2 surface S. For each point $P \in C$ associate a vector $R \in T_P$ such that

$$\frac{dC}{dt} \cdot dN_R = 0 \quad (10)$$

where T_P is the tangent plane of the surface S at the point P and dN_R is the derivative of the normal N of the surface S in the direction R.

Theorem 6 *Inner Surface Constraint: Under orthographic projection, if an image curve C_I is the projection of the curve C on the surface S and $R_I = (r_x, r_y)$ is the projection of the vector R satisfying equation 10, then the change of the orientation, (p,q), of the surface S, along the curve C, in the $p - q$ space is restricted by the image vector R_I, as:*

$$d(p,q)_{C'} \cdot R_I = 0 \quad (11)$$

The proof of the theorem is given in appendix A.

To apply this constraint, we need to identify a curve C in the image plane for which the orientation R can be determined. In a previous paper [Ulupınar and Nevatia, 1990] we have shown that for zero Gaussian curvature surfaces any curve on the surface can be the C curve if the direction R is chosen to be the direction of the rulings of the surface. Following theorem shows how we can use parallel symmetric curves for this purpose in a general case.

Theorem 7 *Let the family of curves,$\{C_i\}$, be on a surface S such that the curves, C_i, are parallel symmetric in 3-D. If the curves C_i are used as the C curves of equation 10 then, the tangent of the curves obtained by joining the symmetric points of the curves C_i gives the direction R of the ISC. Conversely, if the curves obtained by joining the parallel symmetric points of curves, C_i, are used as C curves of equation 10 then the tangents of the curves C_i gives the direction R.*

Proof Consider the parametric representation $S(u, v)$ of the surface S such that the u parameter curves are parallel symmetric to each other (the $\{C_i\}$ family of curves) and v parameter curves join the parallel symmetric points of the u parameter curves.

For the first pat of the theorem we have to show that equation 10 holds or with the current parameterization

$$S_u \cdot N_v = 0 \quad (12)$$

is true, where $N = \frac{S_u \times S_v}{|S_u \times S_v|}$ is the unit normal of the surface. Note that $N \cdot S_u = N \cdot S_v = 0$ by definition. We can substitute $-S_{uv} \cdot N$ for $S_u \cdot N_v$ since:

$$0 = \frac{\partial(S_u \cdot N)}{\partial v} = S_{uv} \cdot N + S_u \cdot N_v \Rightarrow S_u \cdot N_v = -S_{uv} \cdot N \quad (13)$$

S_u is the tangent of the u parameter curves, and since the v parameter curves join the parallel symmetric points of u parameter curves the direction of $S_u(u, v)$ is independent of the v parameter, that is $S_u(u, v) = c(v)S_u(u)$ where c is a scalar function. And;

$$S_{uv} = \frac{\partial S_u}{\partial v} = c'(v)S_u(u) \quad (14)$$

By substituting this in equation 13 we get

$$N_v \cdot S_u = -N \cdot S_{uv} = -c'(v)(N \cdot S_u(u)) = 0 \quad (15)$$

For the second part of the theorem we have to show that $S_v \cdot N_u = 0$. Using equation 15 we get:

$$0 = N_v \cdot S_u = -N \cdot S_{uv} = -N \cdot S_{vu} = S_v \cdot N_u \quad (16)$$

□

3.3 Orthogonality Constraint (OC)

The two previous constraints (CSBC and ISC) are not sufficient to determine surface orientations uniquely. To further constraint the solution, we impose an additional constraint. We require that the cross sections and the meridians of a surface (as defined in sections 2.3 and 2.4) be mutually orthogonal. This constraint may be satisfied precisely for some kinds of surfaces but is not necessarily true for all surfaces; in the latter cases we maximize a measure of orthogonality (given later). This constraint is justified on perceptual observations. It may be viewed as being equivalent to slicing the surface along meridians and cross sections to obtain thin skew symmetric planar patches and assuming that these patches are orthogonally symmetric in 3-D, as in Kanade's analysis for polyhedra [Kanade, 1981]. The orthogonality of two vectors A and B, which lie on a plane having gradient (p, q) and whose images are $A_i = (a_x, a_y)$ and $B_i = (b_x, b_y)$, constrain the gradient (p, q) with the equation:

$$(a_x, a_y, pa_x + qa_y) \cdot (b_x, b_y, pb_x + qb_y) = 0 \quad (17)$$

4 Analysis of ZGC Surfaces

We have applied the constraints of section 3 to analysis of zero-Gaussian curvature surfaces in previous work [Ulupınar and Nevatia, 1990]. We provide a brief summary of this work as the techniques for the PRCGCs and SHGCs are related to it.

A ZGC surface is indicated by the presence of a parallel symmetry and a mirror symmetry where lines of symmetry are straight. The parallel symmetry curves give the cross sections of the ZGC and the lines of symmetry give the rulings. For a ZGC, it is necessary only to consider one cross section at a time, as the surface orientations can simply be propagated along the ruling.

Suppose we wish to estimate the surface orientation at n points along the cross section (assumed to be planar), we have $2n+2$ unknowns ($2n$ for n points, 2 unknowns for the orientation of the cross section itself). ISC and CSBC together provide $2n-1$ constraint equations, leaving three degrees of freedom undetermined. Introducing the orthogonality constraint (in this case requiring the cross section and rulings to be orthogonal) gives an additional n equations; we now have more equations than unknowns.

These equations are, however, not always independent. We find that for a cylindrical surface, all equations can be satisfied exactly and still one degree of freedom remains for the orientation (p_c, q_c) of the cross section plane (it is constrained to be on a line parallel to the axis of the cylinder in the $p-q$ plane). For more general objects, all equations can not be solved exactly. We choose to satisfy CSBC and ISC exactly and minimize a measure of orthogonality. Unfortunately, this minimization procedure also does not, in general, give a unique answer. The minimum is typically achieved when (p_c, q_c) is along a line in the gradient space and the variations are too small along this line to pick a specific value.

This last degree of freedom is removed by using the 3-D shape of the cross section itself. We make the assumption that the 3-D cross section should be as compact as possible, subject to the limits given by other constraints. Our method to accomplish this consists of fitting an ellipse to the cross section and choosing that orientation that gives the least eccentric ellipse in the back projection subject to the orientation satisfying other constraints (namely, its being on a specific line). Also, we apply a correction to this estimate depending on the quality of ellipse fit to bias the answer away from highly slanted orientations. This algorithm is fully described in [Ulupınar and Nevatia, 1990], an outline is given below.

As (p_c, q_c) is constrained to be on a line, the problem is equivalent to estimating only one parameter, say q_c (without loss of generality, as we can rotate the coordinate system as necessary). Steps in estimating q_c are:

1. First Estimation of q_c: An ellipse is fit to the cross section contour, then the orientation of the circle (p_e, q_e), that would project as the fitted ellipse is projected on the q axis, on the $p-q$ plane to obtain the first approximation of q_c, call it q_e.

   It may be necessary to segment the cross section if, it is complex and repetitive. To achieve this, the concavities of the contour are found and matched. If they match in such a way that the cross section is segmented into similar pieces, then a different ellipse is fit to each piece of the contour and average of the ellipses is used to estimate q_e.

2. Updating q_c: The purpose of this updating process is to simulate the bias that humans have in orienting the cross section toward 45^o. We update q_e to obtain the final q_c as follows (after converting q_c into degrees):

$$q_c = 45^o + \lambda(q_e - 45^o) \tag{18}$$

   Where λ is a confidence factor in the range $[0, 1]$ and is a function of how well the ellipse approximates the cross section curve. In our implementation it is given by :

$$\lambda(\epsilon) = (1 - \epsilon^2) \tag{19}$$

   Where ϵ is the ellipse fit error (in range [0,1]).

The algorithm derives from our observations of human perception and we have validated it by an extensive comparison with human subjects.

The described method for recover ZGC surfaces from image contours has been tested on a number of examples (we assume that symmetries are given) and produces results that appear consistent with human observation.

5 Quantitative Shape Recovery of SHGC surfaces

To compute the shape of an SHGC along each recovered cross section curve we can apply the constraints discussed in section 3 as they are applied to a ZGC surface in section 4. For the following; say that there are m cross section curves and we would like to compute the orientation of the surface n points along a cross section. Then we have $2nm$ unknowns, initially, corresponding to the gradient (p, q) of the surface at nm points.

CSBC The curved shared boundary constraint applies between the orientation, (p_c, q_c), of the cross section curves C_j and the orientation, (p_i, q_i) of each of the point on the surface along a cross section. Note that (p_c, q_c) is the same for all cross section curves. The curved shared boundary states that the line in the $p-q$ space from the gradient (p_i, q_i) of a point $P_i \in C_j$ to the gradient (p_c, q_c) of the cross section plane is orthogonal to the tangent, $C_j'(P_i)$, of the cross section C_j at point P_i. Then the constraint equation is:

$$(p_c - p_i, q_c - q_i) \cdot C_j'(P_i) = 0 \quad \forall P_i \in C_j \tag{20}$$

This provides n constraints along each cross section curve.

ISC Inner surface constraint is applied along a cross section using the tangents of the meridians at each point. The theorem 7 indicates that ISC is applicable along the cross section curves because cross section curves are parallel symmetric by theorem 4 with the meridian curves joining the parallel points of the cross section curves. Inner surface constraint states that change of the orientation $(p_{i+1} - p_i, q_{i+1} - q_i)$ of the surface along a cross section curve C_j between two consecutive points $P_i, P_{i+1} \in C_j$ must be orthogonal to the tangent $M_{i+1/2}'(P_{i+1/2})$ of the meridian that passes through the

point $P_{i+1/2} \in C_j$ which is in the middle middle of the points P_i and P_{i+1}. Then the constraint equation is:

$$(p_{i+1}-p_i, q_{i+1}-q_i)\cdot M'_{i+1/2}(P_{i+1/2}) \quad \forall P_i P_{i+1/2} P_{i+1} \in C_j \tag{21}$$

Application of ISC provides $n-1$ equations for each cross section curve.

There are $2n$ unknowns for each cross section curve, and two more unknowns for the whole SHGC, the (p_c, q_c), by combining the two constraints, we have $2n-1$ constraints for each cross section. Then for each cross section there are three degrees of freedom as in the case of a ZGC surface discussed in section 4.

Planarity of Meridians The meridians of an SHGC are planar as discussed in section 2.3. Then the shared boundary constraint can be applied along a meridian curve as if the curve is obtained by cutting the surface of the SHGC with a plane along the meridian. The shared boundary constraint is applied along a meridian, M, between the gradient, (p_m, q_m), of the plane that the meridian M rests on and the gradient (p_j, q_j) of the points $P_j \in M$, using the tangent, $M'(P_j)$ of the meridian curve at each point $P_j \in M$. The constraint equation is :

$$(p_m - p_j, q_m - q_j) \cdot M'(P_j) = 0 \quad \forall P_j \in M \tag{22}$$

Enforcing one meridian curve to be planar automatically makes the others to be planar too. Therefore, the planarity is applied only to one of the meridians, giving m constraint equations with the expense of two additional unknowns.

In total there are now $2nm+4$ unknowns, $2nm$ for the (p,q) of nm points on the surface, two for (p_c, q_c), two more for (p_m, q_m), and there are $2nm$ constraint equations, nm from the CSBC between the cross sections and the face of the surface, $m(n-1)$ from the ISC, and m from the CSBC of a meridian curve. That is there are four degrees of freedom for recovering the orientation of all the points on an SHGC. These four degrees of freedom corresponds to the orientation, (p_c, q_c), of the cross sections and the orientation, (p_m, q_m), of the plane containing the chosen meridian. Without any assumptions we could arbitrarily set these four variables and get a valid reconstruction of the SHGC that would project like the figure in the image plane. However not all of these reconstruction look *natural* to humans when they observe the image of the contours of an SHGC. Humans prefer some interpretations over the others. In the following section we propose orthogonality as the preference criteria.

5.0.1 Orthogonality

For SHGCs we use the orthogonality of the 3-D tangents of the cross sections and the meridian curves, making each little patch, formed by dividing the surface along meridians and the cross sections, orthogonal. We can apply the orthogonality constraint using the equation given in equation 17. This constraint is not always exactly satisfied, except for surfaces of revolution. Therefore we perform a minimization of the second orthogonality constraint as:

$$\Xi = \sum_i \sum_i \cos(\theta_{ij}) = \frac{(C_i(P_{ij}))'_3 \cdot (M_j(p_{ij}))'_3}{|(C_i(P_{ij}))'_3||(M_j(p_{ij}))'_3|} \tag{23}$$

where $(C_i(P_{ij}))'_3$ and $(M_i(P_{ij}))'_3$ are the 3-D tangents of the cross section and meridian curves at point P_{ij}. These 3-D tangents are dependent on their 2-D tangents on the image and on the orientation (p_{ij}, q_{ij}) of the surface at point P_{ij} as given by the equation 17. The gradients (p_{ij}, q_{ij}) at each point is dependent on the four variables, (p_c, q_c) and (p_m, q_m), discussed in the previous section. We would like to minimize the function Ξ for (p_c, q_c) and (p_m, q_m). However from our experiments we observe that minimization of Ξ chooses values that are always consistent with the assumption that the 3-D axis of the SHGC is orthogonal to its cross section.

If we enforce the cross sections to be orthogonal to the axis of the SHGC, the orientation (p_c, q_c) of the cross section lies along a line in the $p-q$ space that passes through the origin and is in the direction of the image of the axis of the SHGC. This constraint also, in effect, enforces the gradient (p_m, q_m) of the plane of the meridians to be orthogonal to the gradient (p_c, q_c) of the cross sections. That is :

$$(p_m, q_m, 1) \cdot (p_c, q_c, 1) = 0 \tag{24}$$

For simplicity, say the coordinate system is rotated such that the image of the axis of the SHGC is aligned with the y axis of the coordinate system. Then, we have; $p_c = 0$ from the orthogonality of the axis to the cross section and $q_m = -1/q_c$ from equation 24. The parameters p_m and q_c are the free variables to be fixed by minimizing the function Ξ. However, the minimum of the function Ξ does not fix the variable q_c (except for surfaces of revolution). Either the function forms a valley along q_c making any choice as good as any other or fixes q_c to be zero which is not a realistic solution. We use the same method for estimating q_c as described for ZGC surfaces in section 4.

5.1 Results

We have implemented the constraints discussed in the previous section in a somewhat reverse order. For an SHGC whose axis is aligned with the y axis of the coordinate system the method is as follows; First the ellipse fit algorithm is applied to compute q_c, then the function Ξ is minimized to compute p_m. Then the surface is constructed using the constraints discussed in section 5 to compute the surface orientation at each point. Figure 11 shows the needle images and the shaded images, with the computed surface orientations, of the SHGCs in figure 1.

6 Quantitative Shape Recovery of PRCGC Surfaces

Here we discuss the application of the three constraints discussed in section 3 along a cross section curve of a PRCGC, to recover the surface orientation of a PRCGC.

CSBC The shared boundary constraint can be applied along the image of a cross section curve. Let (p_c, q_c) be the gradient of the plane that contains the cross section curve, $C(u)$, whose image is the image curve $C_i(u) = (c_x(u), c_y(u))$. Let $(p(u), q(u))$ be the orientation of the points along the cross section curve $C(u)$. Then the

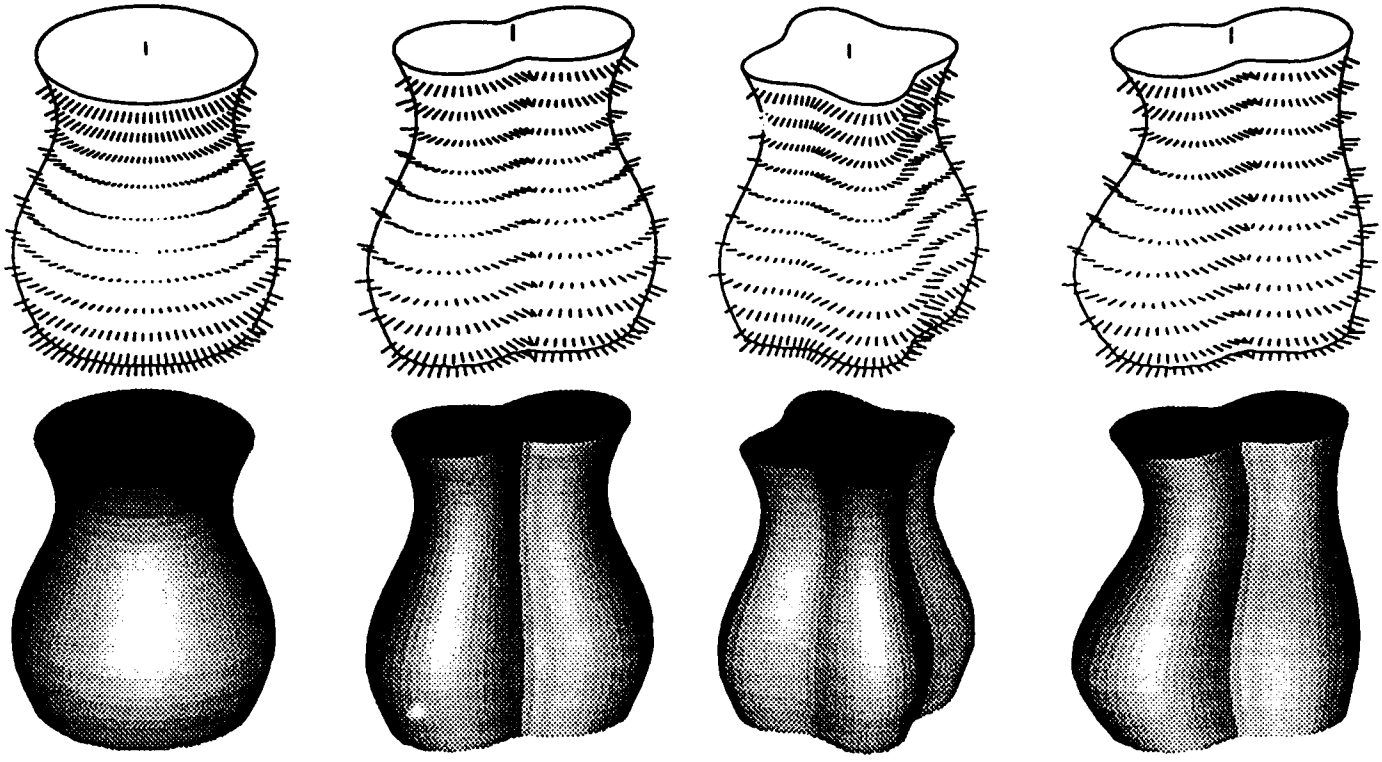


Figure 11: The needle images and the shaded images generated with the computed gradients at each point of the SHGCs in figure 1

shared boundary constraint is:

$$(p_c - p(u), q_c - q(u)) \cdot (c'_x(u), c'_y(u)) = 0 \qquad (25)$$

ISC Theorem 7 indicates that ISC is applicable along the cross sections of a PRCGC because cross sections of a PRCGC join the parallel symmetric points of the meridian curves which are parallel symmetric as given by lemma 1. Since the meridians of a PRCGC are parallel symmetric with cross section curves forming the correspondence, the tangent vectors of the meridians along a cross section is a constant vector which is also parallel to the axis of the PRCGC as given by equation 7. Let the tangent direction of the meridians along the cross section $C(u)$ be A' and its image be $A'_i = (a'_x, a'_y)$, note that A'_i is independent of the u parameter. For the sake of simplicity let us assume that the coordinate system is rotated such that A'_i is along the y axis of the coordinate system, then $a'_x = 0$. The inner surface constraint is :

$$\frac{d}{du}(p(u), q(u)) \cdot (a'_x, a'_y) = 0 \Rightarrow q(u)' a'_y = 0 \Rightarrow q(u) = q_0 \qquad (26)$$

By combining this constraint with the CSBC given in equation 25 we get:

$$p(u) = \frac{c'_y(u)(q_c - q_0)}{c'_x(u)} + p_c \qquad (27)$$

Orthogonality The last constraint is the orthogonality of the meridians to the cross section curves. The reader can easily verify that the u and t parameter curves in equation 3 are orthogonal to each other for all points on the surface S of the PRCGC. Then, we use the orthogonality by enforcing the tangent of the meridians A', whose image is $A'_i = (0, a'_y)$ to be orthogonal to the tangent of the cross section curve C, whose image is $C_i(u) = (c_x(u), c_y(u))$, at a point on the surface whose gradient is $(p(u), q_0)$:

$$(0, a'_y, q_0 a'_y) \cdot (c'_x(u), c'_y(u), p(u)c'_x(u) + q_0 c'_y(u)) = 0 \qquad (28)$$

By substituting $p(u)$ given in equation 27 in the above equation we get:

$$a'_y c'_y(u)(1 + q_o q_c) + a'_y p_c c'_x(u) = 0 \qquad (29)$$

Since the above equation is zero for all values of u we get both $p_c = 0$ and

$$1 + q_o q_c = 0 \Rightarrow q_0 = -1/q_c \qquad (30)$$

Fixing q_c fixes the orientation of the surface along the cross section C together with the gradient (p_c, q_c) (which is $(0, q_c)$ in the rotated coordinate system) of the plane containing C. However our constraint equations do not constrain q_c.

6.1 Recovering Cross Section Curves

In the previous section we have discussed how to recover the surface orientation at each point on a cross section curve given the image of the cross section curve. However it is not directly possible to replicate the images of the cross section curves of a PRCGC, such as the ones in figure 2, except for the ends of the PRCGC, where we assume the cross section curve is given. That is we assume that the surface is cut along its cross sections.

Here we discuss a method for recovering the cross sections when one or both ends of the PRCGC are available, the method also enables us to reconstruct the 3-D PRCGC from the image of it.

At one end of the PRCGC let the image of the end cross section curve be $C_i(u) = (c_x(u), c_y(u))$ and the image of the axis be $A_i(t) = (a_x(t), a_y(t))$ as in the previous section. The image of the axis at the point it intersect the cross section C is $A_i(0) = (a_x(0), a_y(0))$. Say the coordinate system is rotated such that $a'_x(0) = 0$, and the orientation (p_c, q_c) of the plane containing the cross section curve C is computed using the constraints discussed in section 6 with $p_c = 0$. The orientation of the points along the cross section curve C is (p_u, q_0) where $q_0 = -1/q_c$ and $p(u)$ is given by equation 27. Since the meridian curves are parallel symmetric to the axis of the PRCGC we can use the gradient $(p(u), q_0)$ to recover the tangent of the 3-D axis at $t = 0$ as:

$$A'(0) = (a'_x(0), a'_y(0), p(u)a'_x(0) + q_0 a'_y(0)) = (0, a'_y(0), -\frac{a'_y(0)}{q_c}) \equiv (0, q_c, 1) \quad (31)$$

That is $A'(0)$ is parallel to normal, $(0, q_c, 1)$, of the plane containing the cross section C, or the plane Π_a containing $A'(0)$ is orthogonal to the plane of C. Also since the axis, A, of the PRCGC is planar the plane Π_a contains the whole axis curve A.

In the following we give an algorithm for recovering the 3-D cross sections from the image of a PRCGC given the gradient (p_a, q_a) of the plane Π_a containing the axis. Then in the next subsection we give a method for computing (p_a, q_a) from the image.

The gradient (p_c, q_c) of the plane of the cross section C can be computed if the gradient (p_a, q_a) of Π_a is given. The gradient (p_c, q_c) must lie on a line that passes through the origin and in the direction of $A_i(0)$, in our case $p_c = 0$, and $(p_c, q_c, 1)$ is orthogonal to $(p_a, q_a, 1)$ then:

$$(0, q_c, 1) \cdot (p_a, q_a, 1) = 0 \Rightarrow q_c = -\frac{1}{q_a} \quad (32)$$

We can compute the 3-D cross section C from the image C_i of i' by backprojecting C_i to a plane having gradient (p_c, q_c).

If the cross section is rotationally symmetric[2] the algorithm for recovering cross sections is much simpler. In the following we give an algorithm that applies to general, not necessarily rotationally symmetric case.

It can be shown that the image of the axis of the PRCGC, $A_i(t)$, is not always the same as the axis, $B_i(t)$, of the parallel symmetry of the image of the limb edges, where the axis of the PRCGC is the trace of a single point on the cross section as the cross section is swept. This is shown in figure 12. However the image curves $A_i(t)$ and $B_i(t)$ are always parallel symmetric to each other such that the corresponding points are on the same cross section. By using lemma 1 and theorem 5 we conclude that the images of the limb edges are parallel symmetric to each other (and of course to its axis) as well as to the images of the meridians of the surface, and meridians of the surface are parallel symmetric to the axis of the PRCGC by equation 7, so are their images. Therefore the axis of the image of the limb edges, the $B_i(t)$ curve, is parallel symmetric to the image of the axis of the PRCGC, the $A_i(t)$ curve.

[2] A planar cross section is rotationally symmetric iff the lines passing through the center of the cross section intersects both sides of the cross section at equal distances.

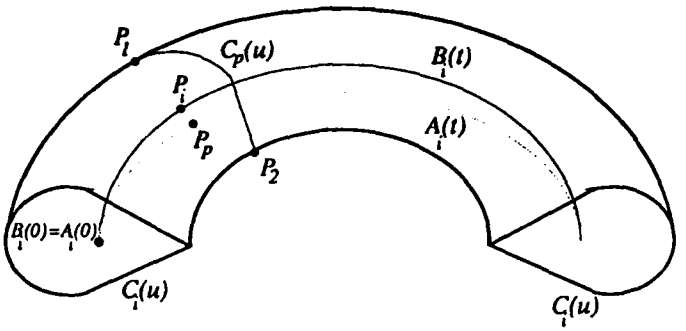


Figure 12: A PRCGC with a non-rotationally symmetric cross section.

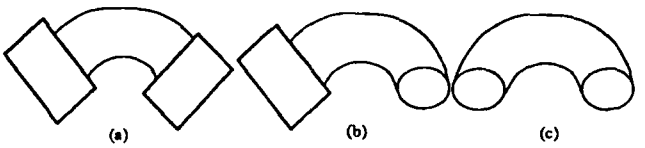


Figure 13: A PRCGC with, (a) none, (b) one, and (c) both end cross sections available.

If we take the axis A of the PRCGC as the trace of the point that is the backprojection of $B_i(0)$ to the cross section plane C. Then $A_i(0) = B_i(0)$. Given the orientation (p_a, q_a) of the plane Π_a containing the axis A, to recover the 3-D cross section say at point P_i on the image axis B_i; The backprojected C of C_i is rotated by the rotation matrix $R(B'(0), B'(P))$ to obtain the 3-D cross section curve $C_p(u)$ at point P, where $B'(0)$ and $B'(P)$ are obtained by backprojecting $B'_i(0)$ and $B'_i(p_i)$ onto the plane Π_a. Then the points P_1 and P_2 that produces the limb edge on the cross section $C_p(u)$ is identified by equating the image tangents of $C_p(u)$ to the image tangent of limb boundaries P_1 and P_2. The position of the cross section C_p in 3-D such that $C_p(P_1)$ and $C_p(P_2)$ project as the points P_1 and P_2 on the image and the point P_p on C_p that corresponds to the point $A(0)$ on C is on the plane Π_a, gives the relative position of the cross section C_p with respect to end cross section C in 3-D.

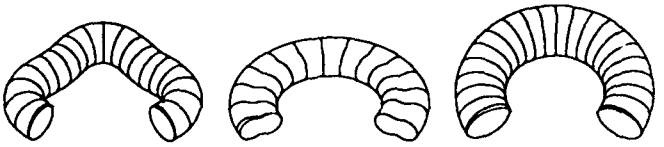


Figure 14: The recovered cross sections for the PRCGCs in figure 2.

6.2 Computing (p_a, q_a)

The gradient (p_a, q_a) of the plane Π_a containing the axis is computed by performing a search in the gradient plane. The objective of the search is to compute (p_a, q_a) that gives a valid reconstruction. A valid construction is one that makes the projection of the cross section points $C_p(P_1)$ and $C_p(P_2)$ exactly the same as the points P_1 and P_2 on the image plane. We form an objective function which is the average distance, on the image plane, of the reconstructed and projected point $C_p(P_2)$ to the point P_2 when $C_p(P_1)$ and P_1 is aligned exactly. Then this objective function is minimized for (p_a, q_a).

The search is facilitated by finding a good initial point for (p_a, q_a) using the shapes of the end cross sections. The analysis in section 6 show that the gradient (p_c, q_c) of the cross section at one end is constraint to be on a line in the gradient space. A particular value on that line may be chosen by using the ellipse fit discussed in section 4. Similar analysis applies to the other end of the PRCGC (if available). Say the orientation of the plane containing the other end cross section C_n is (p_n, q_n). Then the plane of C_n is orthogonal to the plane Π_a. If (p_n, q_n) is not equal to $(0, q_c)$ we can compute an initial normal $N_a = (p_a, q_a, 1)$ of Π_a as $N_a \equiv (p_n, q_n, 1) \times (0, q_c, 1)$. If the other end cross section C_n is not available then the gradient (p_a, q_a) is constrained to be on a line by its orthogonality to $(0, q_c)$. The equation of the line containing (p_a, q_a) is $(0, q_c, 1) \cdot (p_a, q_a, 1) = 0$. Any particular value of (p_a, q_a) may be chosen on this line as the initial (p_a, q_a). Figure 13 shows that perception is more definite when both ends are available, which confirms the above observation that two ends are more informative than one only.

6.3 Results

We have implemented the cross section recovery method described in section 6.1. In the implementation first the orientations (p_c, q_c) and (p_n, q_n) of the end cross sections are computed. Then the normal N_a of Π_a is found by searching around the gradient given by $(p_c, q_c, 1) \times (p_n, q_n, 1)$ that gives a valid reconstruction. The 3-D position of each cross section is then found by translating the end cross section rotating and aligning it with the limb boundaries and the plane of the axis Π_a. Figure 14 shows the recovered cross sections and figure 15 shows the recovered orientations by both needle and shaded images for the PRCGCs given in figure 2.

7 Conclusions

In this paper we have analyzed two class of objects; Straight Homogeneous Generalized Cones (SHGCs) and Planar Right Constant cross section Generalized Cones (PRCGCs).

We show the property of the limb boundaries of SHGCs, under both orthographic and perspective projection, that the tangents of the images of the limb boundaries, if extended from the points on the same cross section, intersect the image of the axis of the SHGC at the same point. We also show that the cross sections of an SHGC are parallel symmetric, and use that property to recover the images of the cross sections of the SHGC. Then we apply the constraints, curved shared boundary constraint (CSBC), inner surface constraint (ISC), and the orthogonality constraint (OC) to the SHGCs. An SHGC has four degrees of freedom if it is to be recovered from the images of its contours without any assumptions. With the assumption of orthogonality there is only one degree of freedom which is fixed by estimating the orientation of the cross sections with an ellipse fit algorithm. Some computational results are shown on synthetic data.

For PRCGCs the limb boundaries are shown to project as parallel symmetric curves, which enable us to find points on the limb boundaries that correspond to the same cross section. We also show that the three constraints, CSBC, ISC and OC, are applicable along the cross section of a PRCGC. We applied the constraints to the ends where the cross sections are available. Then we present an algorithm to reconstruct the 3-D PRCGC from the images of its contours, using the ellipse fit method to recover the orientations of cross sections at the ends.

We have assumed that the object boundaries and symmetries are given. Detection and computation of symmetries may, in itself, be a difficult task in real images. However, we do provide tests that can be used to verify symmetry properties. Also, we believe that 3-D shape recovery process will serve as an aid in segmentation and boundary labelling process as well. In the future, we hope to explore this aspect of the problem.

Appendix

A Proof of the theorem 6

Let $X(u, v)$ be the local parameterization of the surface S around the point $P \in C(t)$ such that for $P = X(u_0, v_0)$, the curve $X(u, v_0)$ is the curve C and the curve $X(u_0, v)$ is in the direction R. That is, u parameter curve is along the curve C and v parameter curve is in the direction R at the point P. Here we have to show that $\frac{d(p,q)}{du} \cdot R_I = 0$ where (p, q) is the normal of the surface in the gradient space, du is in the direction of C', R_I is the image, (x_v, y_v), of the vector $R = X_v = (x_v, y_v, z_v)$ under orthographic projection.

Normal, N, of this surface at any point is given by:

$$N = \frac{X_u \times X_v}{|X_u \times X_v|} \tag{33}$$

Then, the functions dC/dt and dN_R are:

$$\frac{dC}{dt} = \frac{\partial X}{\partial v} = X_v \tag{34}$$

$$dN_R = \frac{\partial N}{\partial u} = N_u \tag{35}$$

By equation 10 we have $X_v \cdot N_u = 0$. Let the normal N of the surface around point P is represented in the (p, q) space as $N = c(p, q, 1)$. Where c is the scale coefficient and equal to $(p^2 + q^2 + 1)^{-1/2}$. Differentiation of N with

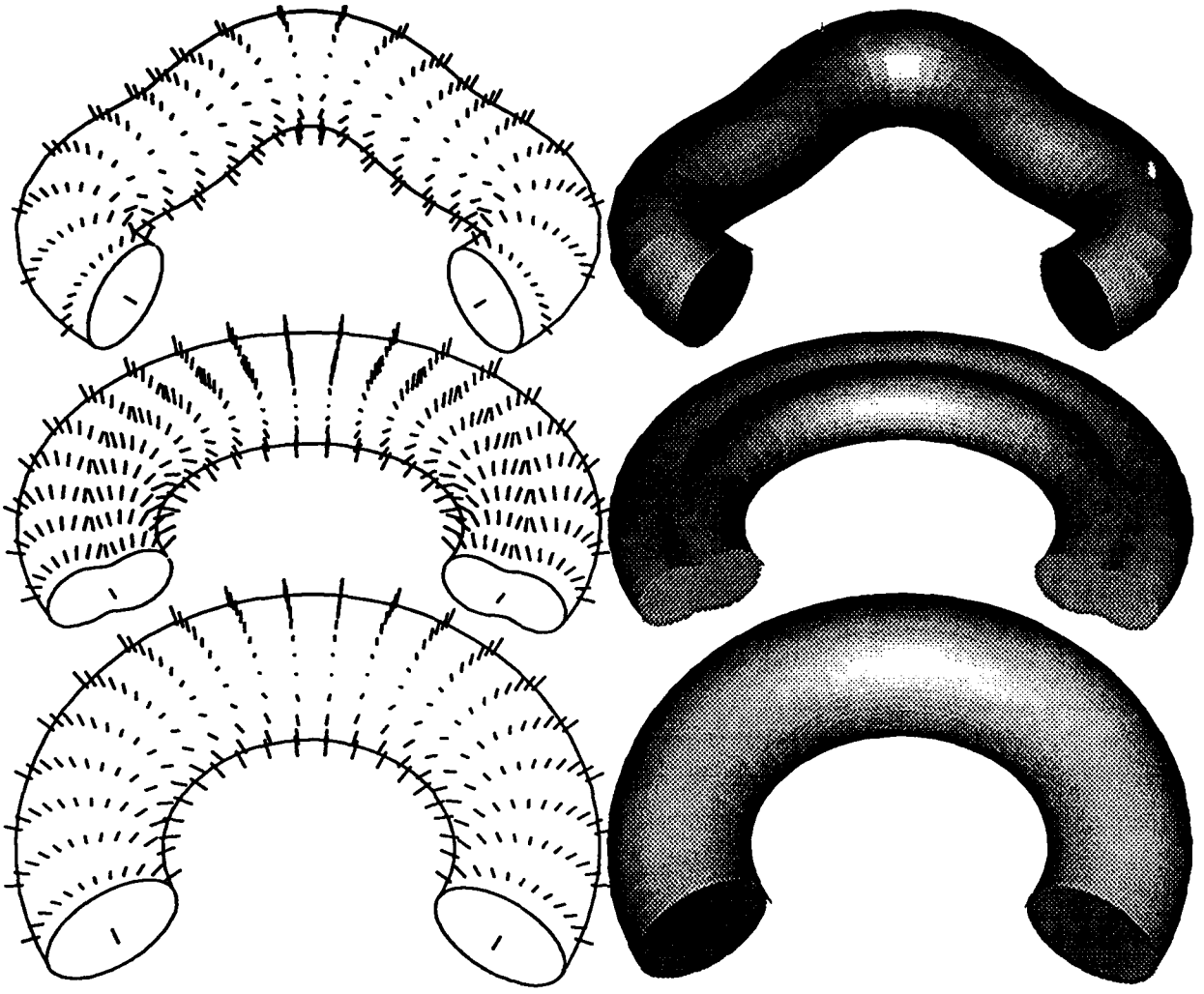


Figure 15: The recovered orientations shown by both needle image and by shading the objects for the PRCGCs in figure 2.

respect to the parameter u gives:

$$\begin{aligned} N_u &= c_u(p,q,1) + c(p_u,q_u,0) \qquad (36) \\ &= \frac{c_u}{c}N + c(p_u,q_u,0) \end{aligned}$$

If we set $X_v \cdot N_u = 0$ where $X_v = (x_v, y_v, z_v)$ and N_u is given in equation 36 we get:

$$X_v \cdot N_u = \frac{c_u}{c} X_v \cdot N + c(x_v, y_v, z_v) \cdot (p_u, q_u, 0) = 0 \quad (37)$$

We also have $N \cdot X_v = 0$ from 33. Therefore

$$\begin{aligned} x_v p_u + y_v q_u &= 0 \\ \frac{d(p,q)}{du} \cdot R_I &= 0 \qquad (38) \end{aligned}$$

□

References

[Barrow and Tenenbaum, 1981] H.G. Barrow and J.M. Tenenbaum. Interpreting line drawings as three dimensional surfaces. *Artificial Intelligence*, 17:75–116, 1981.

[Clowes, 1971] M.B. Clowes. On seeing things. *Artificial Intelligence*, 2(1):79–116, 1971.

[Do Carmo, 1976] M. P. Do Carmo. *Differential Geometry of Curves and Surfaces.* Prentice Hall, 1976.

[Horaud and Brady, 1988] R. Horaud and M. Brady. On the geometric interpretation of image contours. *Artificial Intelligence*, 37:333–353, 1988.

[Huffman, 1971] D.A. Huffman. Impossible objects as nonsense sentences. *Machine Intelligence*, 6:295–323, 1971.

[Kanade, 1981] T. Kanade. Recovery of the three-dimensional shape of an object from a single view. *Artificial Intelligence*, 17:409–460, 1981.

[Koenderink, 1984] J. J. Koenderink. What does the occluding contour tell us about solid shape. *Perception*, 13:321–330, 1984.

[Mackworth, 1973] A.K. Mackworth. Interpreting pictures of polyhedral scenes. *Artificial Intelligence*, 4:121–137, 1973.

[Nalwa, 1987] V. Nalwa. Line drawing interpretation: Bilateral symmetry. In *Proceedings of the Image Un-*

derstanding Workshop, pages 956–967, 1987. Los Angeles.

[Nevatia and Binford, 1977] R. Nevatia and T.O. Binford. Description and recognition of complex-curved objects. *Artificial Intelligence*, 8:77–98, 1977.

[Ponce *et al.*, 1989] J. Ponce, D. Chelberg, and W. B. Mann. Invariant properties of straight homogeneous generalized cylinders and their contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(9):951–966, 1989.

[Rao, 1988] Kashipati Rao. *Shape Description from Sparse and Imperfect Data.* PhD thesis, University of Southern California, 1988.

[Shafer and Kanade, 1983] S.A. Shafer and T. Kanade. The theory of straight homogeneous generalized cylinders. Technical Report Report CS-083-105, Carnegie-Mellon University, 1983.

[Shafer *et al.*, 1983] S. A. Shafer, Kanade T., and J.R. Kender. Gradient space under orthography and perspective. *Computer Vision, Graphics and Image Processing*, 24:182–199, 1983.

[Shafer, 1983] S.A. Shafer. Shadow geometry and occluding contours of generalized cylinders. Technical Report Report CS-83-131, Carnegie-Mellon University, May 1983.

[Stevens, 1981] K. A. Stevens. The visual interpretations of surface contours. *Artificial Intelligence*, 17:47–73, 1981.

[Sugihara, 1986] K Sugihara. *Machine Interpretation of Line Drawings.* MIT Press, 1986.

[Ulupınar and Nevatia, 1988] F. Ulupınar and R. Nevatia. Using symmetries for analysis of shape from contour. In *Proceedings of the 2nd ICCV*, pages 414–426, 1988. Florida.

[Ulupınar and Nevatia, 1990] F. Ulupınar and R. Nevatia. Perception of 3-d surfaces from 2-d contours. In *Proceedings of the 10th International Conference on Pattern Recognition*, pages 147–154, 1990. Atlantic City, New Jersey.

[Xu and Tsuji, 1987] G. Xu and S. Tsuji. Inferring surfaces from boundaries. In *Proceedings of the 1st ICCV*, pages 716–720, 1987. London.

Efficient Parallel Processing in High Level Vision

Craig Reinhart *and Ramakant Nevatia †
Institute for Robotics and Intelligent Systems
Departments of Electrical Engineering and Computer Science
University of Southern California
Los Angeles, California 90089-0273

Abstract

We describe a methodology for developing efficient parallel implementations of high level vision algorithms. Efficiency is defined in terms of algorithm speedup, processor efficiency, system complexity, and programmer burden. Algorithm speedup and processor efficiency are critical issues in the parallel implementation of high level vision tasks as the required algorithms often utilize computationally intensive techniques. Furthermore, due to their usage of complex code and data structures, system complexity and maintenance costs can become excessive if care is not taken in the design of the implementation. Most researchers emphasize speedup and efficiency with little regard to system complexity and programmer burden. We show that through our design procedure, all four issues can be sufficiently addressed.

1 Introduction

Computer vision systems are comprised of tasks that can be categorized into three levels, low, mid, and high. Across the levels, a wide variety of algorithmic techniques are utilized ranging in complexity from simple repetitive processing to elaborate rule-based control structures. Also, the amount of active data at any given point in the system execution can range from tens of thousands of individual scalar values to a few multi-field record structures. Each diverse algorithm utilized in a computer vision system taxes a classical *von Neumann* (serial) architecture in one way or another.

In low-level vision, the multiplications and additions required by convolutional processing are easily executed on a serial machine but the amount of data on which they must operate (the image plane) overwhelm it. Conversely, the small number of abstract data structures utilized in high-level vision are easily maintained by a von Neumann machine but the amount of processing and the complex control structures required to search through a solution space containing permutations of the data soon exceed the capabilities of the machine. To summarize, computer vision systems challenge serial machines through both data intensive and compute intensive operations. These challenges have made parallel implementation of computer vision systems an important topic within the computer vision research community [Ahuja and Swamy, 1984, Weems and Levitan, 1987, Kuehn *et al.*, 1985, Little *et al.*, 1987, Rosenfeld *et al.*, 1986, Hamey *et al.*, 1988, Stout, 1988, Sunwoo and Aggarwal, 1989].

We are interested in investigating the inherent complexities of computer vision systems and how those complexities can be tolerated via *efficient* use of a parallel processor architecture. We define *efficient* in terms of four measures.

Algorithm Speedup is a measure of the reduction in execution time when moving from a sequential to a parallel algorithm implementation. This is a standard measure in the study of parallel processing.

Processor Efficiency, also referred to as load balancing, is a measure of the amount of inherent parallelism, or conversely, the amount of inherent serialism, within an algorithm as well as how well suited the target parallel processor architecture is to the algorithmic requirements. This too is a standard measure in the study of parallel processing.

System Complexity is a measure of how closely the parallel implementation of the algorithm resembles the serial implementation, or conversely, how closely it resembles the parallel processor architecture. This is a measure that we are introducing as it plays an important role in the life cycle of a computer system, both software and hardware.

Programmer Burden is a measure of the degree of difficulty in developing and maintaining the parallel algorithm implementation. This is also a measure that we are introducing as it too plays an important role in the life cycle of a computer system.

An abundance of research into the parallel implementation of low and mid-level vision tasks on a variety of machines has been performed [Rice and Jamieson, 1985, Little *et al.*, 1987, Kuehn *et al.*, 1985, Stout, 1988, Levitan, 1984, Weems, 1988, Hamey *et al.*, 1988] but lit-

*Supported by the Hughes Aircraft Company Fellowship Program

†This research was supported by the Defense Advanced Research Projects Agency, monitored by the Air Force Wright Aeronautical Laboratories under contract F33615-87-C-1436

tle has been done with respect to high-level vision. Furthermore, researchers have placed dramatic emphasis on the issues of algorithm speedup and processor efficiency, especially speedup, with little or no regard to system complexity and programmer burden. Typically, the derived parallel implementations provide good measures of speedup and efficiency at the cost of obscure software and costly, custom built hardware.

In our approach, rather than select a parallel architecture then map an algorithm onto it, as is usually done, we perform some basic analysis steps in order to identify the inherent parallelism contained within the algorithm. We then specify the components of a parallel processor architecture that is well suited to the requirements of the algorithm. For a complete computer vision system comprised of a variety of algorithms, we specify an architecture for each algorithm that is well suited to that algorithm. These architectures can then be realized by either a single heterogeneous or reconfigurable parallel processor architecture. Through this approach we are able to address the issues of system complexity and programmer burden as well as algorithm speedup and processor efficiency.

Due to the need for increased through-put in high-level vision algorithms and the lack of research towards this end as well as the abundance of results available in the parallel implementation of low and mid-level vision algorithms, our studies are centered around high-level vision. In applying our approach to the parallel implementation of a relaxation based image matching algorithm [Medioni and Nevatia, 1984] we were able to:

- Achieve *significant* algorithm speedup.
- Achieve *significant* processor efficiency.
- Design a parallel processor architecture consisting of commercially available components.
- Utilize software that is nearly identical to that used in the serial implementation.

In the following sections we present our methodology for developing parallel implementations of computer vision algorithms and the application of the methodology to the relaxation based image matching algorithm.

2 The Methodology

Research into the parallel implementation of computer vision systems classically begins with the specification of a parallel processor architecture [Little *et al.*, 1987, Stout, 1988, Reisis and Prasanna-Kumar, 1987]. This includes the specification of various organizational parameters such as: *Programming model*, SIMD, MIMD, or MISD [Flynn, 1972]; *Processing elements* (PEs), simple or complex instruction set; *Processing element coupling*, tightly (shared memory) or loosely (message passing); *Processor homogeneity*, homogeneous (identical processing elements) or heterogeneous (two or more different types of processing elements); *Processor synchronization*, synchronous, asynchronous, or loosely synchronous; and *Communication network topology*, cube, mesh, pyramid, ... Details on these and other organizational parameters can be found in [Hwang and Briggs, 1984].

Once a parallel processor architecture has been designed and an algorithm selected, one then proceeds to implement the algorithm on the architecture. This is a two step process. The first step is called the *mapping problem* [Bokhari, 1981] and involves two steps of its own. The second step is development of the actual code. We will not discuss the coding step as it involves the same effort as for a serial algorithm once the mapping problem has been solved.

The mapping problem is solved in two steps, the first involves *partitioning* the algorithm into independent processes and the second, *assigning* the processes to individual processing elements. A formal statement of the problem is: *the search for a correspondence between the interaction pattern of the algorithm processes and the communication network topology of the architecture.* A good solution, or mapping, is one that minimizes the communication overhead and thus maximizes the efficiency and the speedup.

With this approach, if an algorithm is not well suited to the given architecture, the designer is forced into developing an obscure algorithm implementation which resembles the architecture more so than the original algorithm specification.

In our methodology we approach the problem from the opposite direction. That is, we begin by analyzing the algorithm to determine its processing requirements then, using these findings, we specify a parallel processor organization that is well suited to the requirements. We proceed in four basic steps:

- Control Structure Analysis

  In this step we identify the independent processes that constitute the algorithm through inspection of the processing constructs. Of primary interest are iterative constructs (loops) that determine the overall complexity of the algorithm and offer potential for parallelization. This step results in the identification of the inherent parallelism contained within the algorithm.

- Data Structure Analysis

  In this step we determine the data requirements of each process identified above. The result of this step is the specification of which data structures to partition and how to partition them (distribute them among processes.)

- Communication Analysis

  Identification of the independent processes and the data structure partitioning scheme will determine the communication requirements between the processes. That is, a data structure may be distributed among processes such that one process is assigned a data item required by another process to complete its task. In this step such requirements are determined as well as the appropriate communication protocols for their implementation, such as synchronous message passing among all processes, asynchronous message exchanges between two processes, message broadcasting and reduction. The result of this process will lead to the specification of

the communication network topology of the architecture.

- Architecture Specification

  Given the results of the previous steps, this step is where we specify the architecture in terms of its organizational parameters. The result is the specification of a parallel processor architecture well suited to the requirements of the specified algorithm in terms of speedup, efficiency, system complexity, and programmer burden.

We have found that this approach produces high degrees of speedup and efficiency via software that resembles the serial implementation of the algorithm and is therefore no more difficult to develop and maintain. Furthermore, this approach lends itself to the design of parallel implementations of complete computer vision systems (heterogeneous algorithm suites) which can be implemented via a reconfigurable or a heterogeneous parallel processor architecture.

3 Image Matching – An Application

3.1 Overview

Matching of two images (or a map and an image) is a fundamental operation in computer vision. Various solutions to the problem of finding correspondences between images have been proposed ranging from correlation [Rosenfeld and Kak, 1976] to graph isomorphism [Ghahraman *et al.*, 1980]. One primary distinction among the proposed solutions is the level of description at which the matching is performed. Correlation based techniques typically operate directly on sensor data (pixels) whereas graph based approaches often utilize semantic structures such as roads and buildings.

The image matching algorithm used in our study utilizes a discrete relaxation based approach to matching. It determines correspondences between line segments detected in each image based on symbolic descriptions of the segments as well as geometrical relationships between segments. The algorithm iterates over the solution space until a stable state is converged upon.

This algorithm was selected for study due, primarily, to its applicability to high-level vision. But, the basic approach utilized in the algorithm, relaxation, has been used in other applications as well [Waltz, 1972, Rosenfeld *et al.*, 1976, Faugeras and Price, 1981, Rosenfeld and Smith, 1981, Terzopoulos, 1986, Rutkowski *et al.*, 1981]. Therefore, the results of this study can be generalized to various other applications in low, mid, and high-level vision.

In the following sections we present details of the application of our methodology to the relaxation based image matching algorithm. We present a brief description of the algorithm, application of the four steps that constitute our methodology, and a discussion of the results of the application in terms of our four measures, algorithm speedup, processor efficiency, system complexity, and programmer burden.

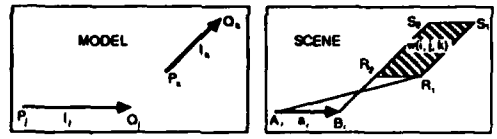


Figure 1: Window construction.

3.2 Algorithm Description

This image matching algorithm [Medioni and Nevatia, 1984] receives input images from two independent sources and then attempts to construct a list of correspondences between them using a relaxation based approach. We provide an overview of the algorithm with enough detail to discuss our algorithm mapping methodology. For details and explanations beyond the scope of our discussion, the reader should see the referenced work.

The primitives used by the image matching algorithm are *linear segments*, represented symbolically by their end point coordinates, orientation, and average contrast. Given two sets of linear segments extracted from two images (or an image and a map), the object is to find correspondences between the segments of each set based on their symbolic descriptions (local constraints) and on the geometric relationships between segments of the same image (global constraints.) The assumptions made prior to matching are that: 1) the orientations of the two images are nearly the same; and 2) the scaling factor from one image to the other is approximately known.

The set of primitives, $A = \{a_i | 1 \leq i \leq n\}$, from one image is called the *SCENE* and the primitives, a_i, are called *OBJECTS*. The set of primitives, $L = \{l_j | 1 \leq j \leq m\}$, from the other image is called the *MODEL* and the primitives, l_j, are called *LABELS*. The algorithm proceeds to compute the quantity $p(i,j)$ in $\{0,1\}$, which is the *POSSIBILITY* that object a_i corresponds to label l_j. It is possible that an object has no corresponding label due to occlusion or scene change, that several objects correspond to the same label due to fragmentation, or that an object corresponds to several labels due to merging. The method for computing $p(i,j)$ relies on geometrical constraints, that is, when a label, l_j, is assigned to an object, a_i, we expect to find an object, a_h, with a label, l_k, in an area defined by i, j, and k. The area is called a *WINDOW* and is denoted $w(i,j,k)$.

The method for computing $w(i,j,k)$ is as follows. The object, a_i, is represented by the two dimensional vector $\vec{A_iB_i}$ and the label, l_j, by $\vec{P_jQ_j}$. By "sliding" l_j over a_i an area is described by the corresponding motion of label l_k, $\vec{P_kQ_k}$ (figure 1.) This parallelogram shaped area is the window $w(i,j,k)$.

Two object/label assignments, (i,j) and (h,k), are *COMPATIBLE*, $(i,j)C(h,k)$, if and only if object a_h lies within window $w(i,j,k)$ and object a_i lies within window $w(h,k,j)$.

Using these definitions, the algorithm searches for object/label correspondences by first identifying all possible correspondences based on the symbolic descriptions of the objects and labels. This set of correspondences

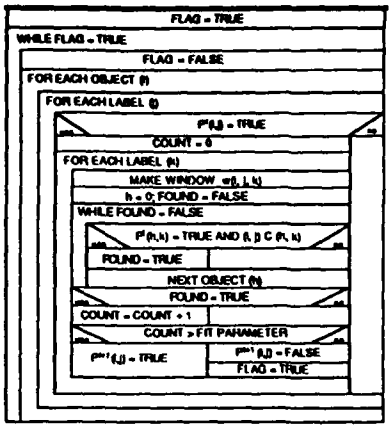


Figure 2: Image matching algorithm primary flow.

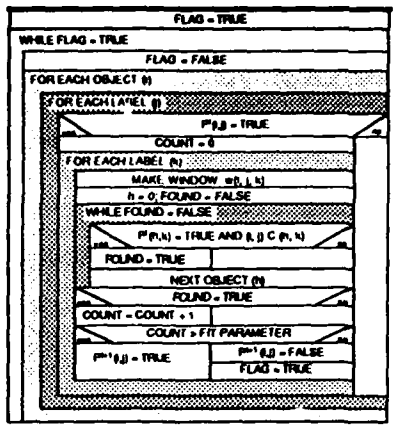


Figure 4: Image matching primary control loops.

```
flag = 1;
while (flag) {
  flag = 0;
  /* Iteration of possibilities/compatibilities. */
  for (i = 0; i < number_of_objects; ++i)
    for (j = 0; j < number_of_labels; ++j)
      if (p[j][i]) { /* if P(t)(i,j) == 1 */
        card_s = 0;

        /* Compute the degree of support for */
        /* the object/label assignment. */

        for (k = 0; k < number_of_labels; ++k) { /* for all labels */
          make_window(objects[i], labels[j], labels[k], win_ijk);
          h = 0; found = 0;
          while ((h < number_of_objects) && (!found)) {
            if (p[k][h] && in_window(objects[h], win_ijk)
              found = compatible(objects[i], labels[j], objects[h], labels[k]);
            ++h;
          } /* while ((h < number_of_objects) ... */
          if (found)
            ++card_s;
        } /* for (k = ... */

        if (card_s < q) {
          flag = 1;
          p[j][i] = 0;
        } /* if (card_s ... */
      } /* if (p[j][i] ... */
  } /* while (flag) ... */
```

Figure 3: Serial code for image matching algorithm.

constitutes the possibilities at iteration step 0, $p^0(i,j)$. Subsequent values of $p(i,j)$ are computed by the *iteration formula*:

$$\forall (i,j), p^{t+1}(i,j) = 1 \text{ if } p^t(i,j) = 1 \text{ AND}$$
$$\exists \text{ subset } S \text{ of } [1,m] \text{ (labels) with } q \text{ elements such that}$$
$$\forall s \text{ in } S, \exists k \text{ in } [1,n] \text{ (objects) such that } p^t(k,s) = 1 \text{ and}$$
$$(i,j)C(k,s).$$

The algorithm halts when $\forall (i,j), p^{t+1}(i,j) = p^t(i,j)$.

The value q is the *fit parameter*. If a perfect match is desired then its value should be set to m, the number of labels. Otherwise it should be set to a value determined by the desired degree of match between the two images. A flow diagram of the image matching algorithm is provided in figure 2 and a code segment from the serial implementation in figure 3.

3.3 Control Structure Analysis

In analyzing the control structure of an algorithm our objective is to determine its overall time complexity and to identify the specific structures that dictate that time complexity, typically loop constructs. We call these constructs *primary control structures*. Identification of the primary control structures will help us to identify independent processes and thus, identify areas where parallelism can be applied providing significant algorithm speedup.

The time complexity of the image matching algorithm is determined as follows. Given a scene containing n objects and a model containing m labels, the maximum number of possible object/label pairs is nm, which occurs when every object is similar to every label. At each iteration at most one object/label pair is discarded, that is, its possibility is set to 0, therefore, the process converges in at most nm iterations. During each iteration the algorithm computes the possibility of the object/label pair which is a measure of how well it 'fits' with the remaining object/label pairs. In the worst case, this requires investigating nm pairs. Therefore, the complexity of the algorithm is $O(n^2m^2)$. If we assume an equal number of objects and labels, m, the algorithm time complexity can be expressed as $O(m^4)$. Figure 4 shows, pictorially, the four nested loops which implement this time complexity. These constitute the primary control structures.

Nested within the four loops is the *possibility computation*. As described above, it consists of checking whether or not a given object/label pair has any *compatible* object/label pairs. This, in turn, requires the computation of a window and the search for an object within it. Once a candidate object/label pair, (a_i, l_j), has been queued, the possibility computation, $p(i,j)$, can proceed as m^2 independent computations. Each computation is structured so that it operates on an isolated data set, that is, successive passes through the inner loops (the possibility computation) are independent of one another. Thus, the possibility computation can proceed as multiple paralliel processes and has the potential to provide significant al-

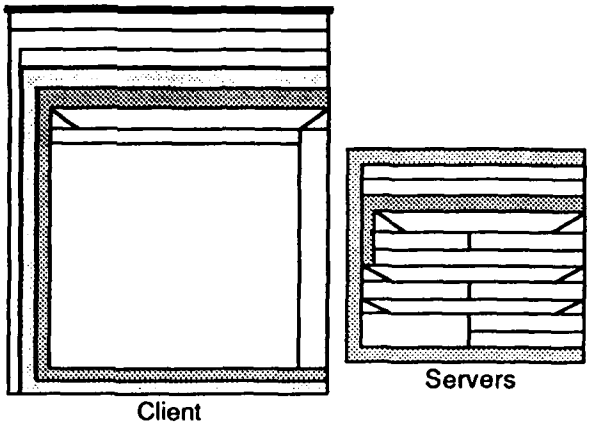


Figure 5: Client/Server algorithm partitioning.

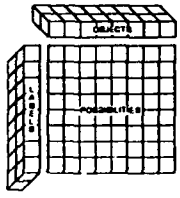


Figure 6: Image matching primary data structures.

gorithm speedup. For these reasons, it constitutes our process partitioning scheme.

Having selected the possibility computation as the process with which to partition the algorithm, we have produced a client/server model. That is, one process will queue possible object/label pairings via the outer two primary control loops, constituting the client, and a set of independent processes will determine the possibility of that pairing via execution of the inner two control loops and their encompassed procedures in a distributed fashion, constituting the servers. Figure 5 shows the client/server algorithm partitioning.

3.4 Data Structure Analysis

Having identified the possibility computation as the task on which to partition the algorithm into processes, we must now determine the data requirements of each computation. In doing so we will identify the *primary data structures* and determine an appropriate partitioning of these structures.

For the image matching algorithm, three primary data structures can be identified. The first two are linear arrays of size m of symbolic records, one array each for storage of the set of objects and the set of labels. The third data structure is an mxm matrix of logical values that store the results of the possibility computation, $p^t(i,j)$, for each iteration, t. Figure 6 shows the primary data structures, pictorially.

Each possibility computation (process) requires two entries from the object array, a_i and a_h, and two entries from the label array, l_j and l_k. All processes receive

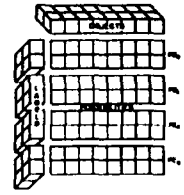


Figure 7: Image matching horizontal swath partitions.

the same (a_i, l_j) pair, the object/label assignment under consideration, and each receives a unique (a_h, l_k) pair, an object/label assignment that determines the global consistency of the pair under consideration. From these inputs the windows, $w(i,j,k)$ and $w(h,k,j)$, are formed. The relation $(i,j)C(h,k)$ is then computed by determining whether or not a_h lies within $w(i,j,k)$ and a_i lies within $w(h,k,j)$. A value of 1 is returned if the relation holds, otherwise a value of 0 is returned. The value of $p^{t+1}(i,j)$ is determined by summing the results from all of the individual processes and comparing that sum to the fit parameter, q.

If we assume the availability of $N = m^2$ processing elements, the obvious way of partitioning the data structures is to assign each PE, $0 \leq p \leq N-1$, an object/label pair, $(a_h, l_k) \in AxL$. If the number of processing elements available is less than m^2, that is, $N \ll m^2$, then the most intuitive way, from a programmer's viewpoint, to partition the data structures is to assign each PE, $0 \leq p \leq N-1$, to a $1/N$ sized portion of the label array and the entire object array thus giving each a set $S_p = \{(a_h, l_k) | 1 \leq h \leq m, p*(m/N) \leq k \leq p*(m/N) + m/N - 1\} \forall p : 0 \leq p \leq N-1$ of objects and labels. This creates N horizontal swathes through the possibility matrix as depicted in figure 7. These horizontal swathes constitute our data partitioning scheme.

3.5 Communication Analysis

Having designed our process and data partitions, we must now identify the inter-process communication required to complete the parallel implementation.

As described previously, a possibility computation requires access to the object/label pair under consideration, (a_i, l_j), provided by the client process, and the set of possible object/label pairs from which the server processes compute a degree of support. The set of possible pairs are statically distributed among the server processes once, upon algorithm initiation, as described above. Conversely, the pair (a_i, l_j) must be provided to each server, dynamically, by the client process. This is achieved via a *broadcast* operation from the client to every server.

Having received (a_i, l_j), each server process computes a degree of support for the pair based on its set of possible object/label pairs (its data partition.) Upon completion, each server reports its degree of support to the client where the individual degrees of support are combined into a single result and the possibility computation, $p^t(i,j)$, is completed. This is achieved via a *reduction* operation from every server to the client.

Finally, the client must report $p^t(i,j)$ to the server process whose data partition includes the pair (a_i, l_j) so that it can update its possibility value. This is achieved via a point-to-point *send/receive* operation from the client to the particular server.

In summary, our process/data partitioning scheme requires three types of communication: 1) broadcast; 2) reduction; and 3) point-to-point send/receive.

This concludes the analysis steps of our methodology as applied to the image matching algorithm. We have described the algorithm, identified its primary control structures, identified its primary data structures, partitioned it into independent processes, and identified all required inter-process communication. Our remaining task is to specify a parallel architecture well suited to the requirements identified by our analysis. This is presented in the following section. We then present an evaluation of the system design arrived at via our methodology through architecture simulation and actual implementation.

3.6 Architecture Specification

In specifying a parallel processor architecture we must address various organizational parameters: *Programming model*; *Processing element type*; *Processing element coupling*; *Processor homogeneity*; *Processor synchronization*; and *Communication network topology*. Whereas in the classical approach this is done prior to the algorithm analysis, that is, the parallel implementation of the algorithm is specified for a particular parallel architecture, we base our specification of these parameters on the results of our algorithm analysis. In the following paragraphs we address each of these organizational parameters and discuss how they are influenced by the processing requirements of the image matching algorithm.

Programming Model. The image matching algorithm (more specifically, the possibility computation) contains various processing steps that are data dependent, that is, all data items are not processed identically. The *M*ultiple *I*nstruction *M*ultiple *D*ata programming model is best suited to this situation. In this model each processing element can execute code dictated by its particular data items. Conversely, the algorithm could be implemented under the *S*ingle *I*nstruction *M*ultiple *D*ata programming model, as demonstrated in [Reisis and Prasanna-Kumar, 1987], but processing elements would spend a great deal of time "idling" through code which is not applicable to their data items and thus, reduce the processor efficiency.

Processing Element Type. Computation of the compatibility relationship, $(i,j)C(h,k)$, between to pairs of object/label correspondences, (a_i, l_j) and (a_h, l_k), requires computation of two windows, $w(i,j,k)$ and $w(h,k,j)$, as well as whether or not the objects a_i and a_h lie within the respective windows. These computations require the use of transcendental functions as well as floating point arithmetic (unless integerization is performed.) Therefore, the processor utilized must support these computations. Furthermore, to reduce system complexity and programmer burden, the processor must be programmable in a high-level language that allows specification of the primary data structures in a natural way, that is, via multi-field records. Processors best suited to these constraints are of the complex instruction set variety such as a general purpose microprocessor.

Processing Element Coupling. As the communication between processes is in bursts, that is, at the beginning of each possibility computation (the broadcast) and at the end of each possibility computation (the reduction), a tightly coupled or shared memory system would not suffice because of memory access conflicts. Without special protocols to allow concurrent reading and writing of memory, a communication bottle neck would exist. Better suited to the algorithm is a loosely coupled or message passing architecture. These systems facilitate high bandwidth communication without the requirement of special purpose hardware.

Processor Homogeneity. Our partitioning scheme provides each server process with identical tasks. The client process is computationally similar to the server processes in that it utilizes the same data structures as well as similar logic. Therefore, the parallel architecture should be homogeneous, that is, comprised of a set of identical processing elements. This facilitates programming (reduction of programmer burden) as well as hardware interfacing of processing elements (reduction of system complexity.)

Processor Synchronization. In light of the fact that there is computational similarity between all of the identified processes as well as data dependent processing, the parallel architecture should operate in loosely synchronous mode. That is, all processes incorporate identical copies of the program, with the exception of the client process, and execute under control of their own program counter. Synchronization occurs only at points of communication. As we shall see, this also facilitates programmability of the implementation which reduces system complexity and programmer burden.

Communication Network Topology. Perhaps the most interesting aspect of a parallel processor architecture is its communication network topology, the processing element interconnect pattern. As we showed via the communication analysis, the image matching algorithm places three constraints on the communication network topology. The first is that it must facilitate an efficient broadcast operation, the second is that is must facilitate an efficient reduction operation, and the third is that it must facilitate an efficient point-to-point send/receive operation. In the following paragraphs we consider each of these constraints.

With regard to the broadcast operation, the ideal message passing architecture is one containing a single common bus to which all processing elements are connected. In this topology a broadcast operation is completed in $O(1)$ time.

With regard to the reduction operation, the ideal algorithm requires $\Omega(\log n)$ time, that is, "order no less than $\log n$ time", assuming concurrent read and write operations are forbidden [Cole and Vishkin, 1986]. This ideal time is achieved by an algorithm that utilizes a divide and conquer approach. The result is obtained by dividing the data set into two halves, finding the two partial

results, and combining the partial results to get the final result. The dividing is done recursively until the data sets are indivisible. Such a divide and conquer scheme yields a binary tree with n^2 nodes with the data items starting at the leaves. For the image matching algorithm the data items, the objects and labels used to determine the global validity of a queued object/label correspondence, can be distributed among all nodes of the tree, not just the leaves.

With regard to the point-to-point send/receive operation, the ideal message passing architecture is, again, one containing a single common bus to which all processing elements are connected. In this topology a send/receive operation is completed in $O(1)$ time.

The reduction operation produces the most stringent constraint dictated by the image matching algorithm. A communication network topology that facilitates this operation will also facilitate the other two as they are of lower order complexity. Therefore, for parallel implementation of the image matching algorithm, the processing elements should be connected via a binary tree topology.

To summarize, the organizational parameters of a parallel processor architecture that is well suited to the image matching algorithm should be specified as follows:

- Programming model – MIMD
- Processing elements – Complex Instruction Set Computers
- Processor coupling – Loosely Coupled
- Processor homogeneity – Homogeneous
- Processor synchronization – Loosely Synchronous
- Communication network topology – Binary Tree

This completes the application of our methodology to the image matching algorithm. In the next section we present an evaluation of the system design in terms of our measures; algorithm speedup, processor efficiency, system complexity, and programmer burden.

3.7 System Evaluation

Having completed our parallel implementation of the image matching algorithm, we now present an evaluation of the implementation in terms of our four measures. The evaluation is performed on the basis of three "data points." First, we use a serial implementation of the algorithm as a baseline with which comparisons can be performed. Second, we use a simulation developed to analyze the implementation relative to any number of processing elements. Third, we use an actual system implementation utilizing INMOS Transputers [INM, 1989] to bring validity and feasibility to the entire study.

As we stated earlier, most research in the field of parallel processing of computer vision algorithms is primarily concerned with algorithm speedup and processor efficiency. For this reason we begin our evaluation and discussion with these two measures.

Algorithm speedup is defined as the ratio of elapsed time when executing a program on a single processor to the elapsed time when N processors are available. That is, for N processing elements, algorithm speedup is defined as

$$S_N = \frac{T_1}{T_N}$$

where T_1 and T_N are the elapsed times for 1 and N processing elements, respectively.

Processor efficiency is defined as the average utilization of the available processing elements and can be specified in terms of algorithm speedup, S_N. For N processing elements, processor efficiency is defined as

$$E_N = \frac{S_N}{N}.$$

If the efficiency, E_N, of a parallel implementation remains constant (ideally 1) as the number of processing elements, N, is increased, the parallel implementation of the algorithm is said to have achieved *linear speedup*.

3.7.1 Complexity Analysis

Previously we determined the complexity of the image matching algorithm to be $O(m^4)$, assuming an equal number of objects and labels, m. This is due to the nested loop structure of the algorithm where every object/label pair, (a_i, l_j), is checked against every other object/label pair, (a_k, l_k) for compatibility.

In our partitioning strategy, we distribute the m^2 compatibility computations for each object/label pair possibility computation evenly among the N processing elements. Therefore, barring the existence of any data dependencies or overhead, we expect to achieve $O(N)$ speedup and complete processor utilization, that is, an efficiency of 1. Unfortunately, both data dependencies and overhead exist.

The data dependencies contained within the image matching algorithm can be expressed in terms of the possible correspondences between objects and labels. Let us define the value P_j to be the set of possible object correspondences for each label l_j, $1 \leq j \leq m$. We can then define

$$\hat{P} = \max_{j=1}^{m} |P_j|,$$
$$\bar{P} = \frac{\sum_{j=1}^{m} |P_j|}{m}, \text{ and}$$
$$k = \frac{\hat{P}}{\bar{P}}.$$

The value k is an indication of how evenly the object/label correspondences are distributed. For instance, if every label forms possible correspondences with the same number of objects, k will be 1. Conversely, if one label forms possible correspondences with a large number of objects and the remaining labels form possible correspondences with a small number of objects, then k will be large.

Using these definitions, the *expected* values for algorithm speedup and processor efficiency for our implementation of the image matching algorithm (barring any overhead) are

$$S_N^e = \frac{N}{k} \text{ and}$$
$$E_N^e = \frac{1}{k}.$$

As an appeal to one's intuition, consider the following cases. Recall that our data partitioning scheme calls for the assignment of a set of object/label pairs to each processing element. If all sets contain an equal number of possible correspondences, then

$$\hat{P} = \bar{P} \Rightarrow k = 1 \Rightarrow$$
$$S_N^e = N \text{ and } E_N^e = 1.$$

This implies that each processing element is assigned the same amount of work. If one set contains more possible correspondences than all of the rest, $\exists j : |P_j| \gg |P_i| \forall i \neq j, 1 \leq i \leq m$, then

$$\hat{P} \gg \bar{P} \Rightarrow k \gg 1 \Rightarrow$$
$$S_N^e \ll N \text{ and } E_N^e \ll 1.$$

This implies that the processing element assigned the set P_j must do more work than any of the other processing elements.

These expected values are to be considered estimates of the overhead incurred by the implementation due to data dependencies. One must remember that the actual distribution of possible object/label correspondences is dynamic as it is the goal of the algorithm to reduce this to a canonical set of correspondences via the relaxation operation. Furthermore, the algorithm contains some inherently serial operations, those of the client process, that must be taken into consideration in light of the overall performance analysis. Actual values of algorithm speedup and processor efficiency will vary due to this dynamic behavior and serialism. The goal is to minimize the effects of these on the performance of the parallel implementation.

3.7.2 Measured Performance

To measure the actual values of algorithm speedup and processor efficiency we devised three test cases. The first is comprised of two identical images containing multiple vertical lines. In this scenario $k = 1$. The second is comprised of an image containing one vertical line and multiple horizontal lines and an image containing one horizontal line and multiple vertical lines. In this scenario $k = m/2$ where m is the number of labels. The third is comprised of two identical images containing lines extracted from an airfield image. In this scenario $k = 1.67$.

Table 1 shows the execution times for the three scenarios when instantiated with various problem sizes. The first four rows are for the first scenario with the number of labels, m, being 12, 24, 36, and 48. The next three rows are for the second scenario with the number of labels being 50, 100, and 200. The last three rows are for the third scenario with the number of labels being 51, 102, and 153. Simulation runs were done with the number of processing elements being 1, 2, 3, 4, 15, and m, the number of labels. These are represented by the six columns.

Table 2 shows the measured speedup for each of the test cases and table 3 shows the efficiency. One should note that these measured values do not include overhead for inter-processor communication. They strictly reflect the algorithm speedup and processor efficiency as affected by our process and data partitioning schemes and the data dependencies.

To observe the effects of inter-processor communication on the overall performance (as well as to demonstrate a complete application of our methodology) we also developed an actual parallel processor system based on our implementation of the image matching algorithm.

Problem	T_1	T_2	T_3	T_4	T_{15}	T_m
12	10.53	5.22	3.73	3.13	–	1.0
24	167.57	97.08	63.23	53.43	18.83	9.3
36	870.90	473.82	342.62	271.02	90.52	31.1
48	2743.32	1487.00	1057.93	792.52	283.43	72.7
50	6.43	3.45	2.37	1.78	0.58	0.2
100	33.60	18.10	12.13	9.15	2.75	0.5
200	271.80	140.72	95.78	68.75	19.15	1.8
51	75.07	39.37	28.67	21.78	6.82	2.1
102	1091.93	552.42	375.27	300.35	83.28	18.3
153	6620.68	3396.95	2267.65	1700.68	547.67	78.0

Table 1: Execution times from simulation.

Problem	S_1	S_2	S_3	S_4	S_{15}	S_m
12	1.00	2.02	2.82	3.36	–	9.75
24	1.00	1.73	2.65	3.14	8.90	18.02
36	1.00	1.84	2.54	3.21	9.62	27.96
48	1.00	1.84	2.59	3.46	9.68	37.73
50	1.00	1.86	2.71	3.61	11.09	27.96
100	1.00	1.86	2.77	3.67	12.22	58.95
200	1.00	1.93	2.84	3.95	14.19	145.35
51	1.00	1.91	2.62	3.45	11.01	34.59
102	1.00	1.98	2.91	3.64	13.11	59.67
153	1.00	1.95	2.92	3.89	12.09	84.86

Table 2: Speedup from simulation.

The system is comprised of one to four INMOS Transputers but is expandable to incorporate any number of processing elements without any system redesign.

Tables 4, 5, and 6 show the measured execution times, algorithm speedup, and processor efficiency, respectively, for the various test cases and problem sizes.

Although our data points are sparse, the tables do indicate the following trends in terms of algorithm speedup and processor efficiency for our parallel implementation of the image matching algorithm:

- The implementation is most effective when the problem size is large, that is, when the number of possible object/label correspondences is large.
- The implementation is most effective when the number of processing elements is less than or equal to

Problem	E_1	E_2	E_3	E_4	E_{15}	E_m
12	1.00	1.00	0.94	0.84	–	0.81
24	1.00	0.86	0.88	0.78	0.59	0.75
36	1.00	0.92	0.85	0.80	0.64	0.78
48	1.00	0.92	0.86	0.87	0.65	0.79
50	1.00	0.93	0.90	0.90	0.74	0.55
100	1.00	0.93	0.92	0.92	0.81	0.58
200	1.00	0.97	0.95	0.99	0.95	0.73
51	1.00	0.95	0.87	0.86	0.73	0.68
102	1.00	0.99	0.97	0.91	0.87	0.58
153	1.00	0.97	0.97	0.97	0.81	0.55

Table 3: Efficiency from simulation.

Problem	T_1	T_2	T_3	T_4
12	8	5	4	3
24	119	71	50	40
36	581	338	238	185
48	1815	1060	747	582
50	7	5	4	4
100	31	23	19	18
200	159	106	83	78
51	60	34	23	21
102	839	439	303	233
153	4063	2112	1373	1080

Table 4: Execution times from Transputer implementation.

Problem	S_1	S_2	S_3	S_4
12	1.00	1.60	2.00	2.67
24	1.00	1.68	2.38	2.98
36	1.00	1.72	2.44	3.14
48	1.00	1.71	2.43	3.19
50	1.00	1.26	1.52	1.54
100	1.00	1.36	1.63	1.73
200	1.00	1.50	1.91	2.05
51	1.00	1.76	2.40	2.86
102	1.00	1.91	2.77	3.60
153	1.00	1.92	2.96	3.76

Table 5: Speedup from Transputer implementation.

the number of labels (when data dependencies are taken into account.)

- In light of the previous items, inter-processor communication does not dominate the implementation.

We now focus our attention on our two new measures, system complexity and programmer burden.

3.7.3 System Development and Maintenance

As stated earlier, system complexity is a measure of how closely the parallel implementation of the algorithm resembles the serial implementation. It can also be viewed as the amount of effort (cost) required to realize the parallel implementation of the algorithm.

Previously, we showed a program segment for the image matching algorithm's primary control structures as

Problem	E_1	E_2	E_3	E_4
12	1.00	0.80	0.67	0.67
24	1.00	0.84	0.79	0.76
36	1.00	0.86	0.81	0.79
48	1.00	0.86	0.81	0.86
50	1.00	0.63	0.51	0.39
100	1.00	0.68	0.54	0.43
200	1.00	0.75	0.64	0.51
51	1.00	0.88	0.80	0.72
102	1.00	0.96	0.92	0.90
153	1.00	0.96	0.99	0.94

Table 6: Efficiency from Transputer implementation.

```
flag = 1;
while (flag) {
  flag = 0;
  /* Iteration of possibilities/compatibilities. */
  for (i = 0; i < number_of_objects; ++i)
    for (j = 0; j < number_of_labels; ++j)
      if (p[j][i]) { /* if P(t)(i,j) == 1 */
        card_s = 0;

        SEND OBJECT/LABEL ASSIGNMENT PAIR TO CHILDREN
        (BROADCAST).

        /* Compute the degree of support for the object/label */
        /* assignment provided by this PEs 1/nth of the label table. */
        for (k = 0; k < my_share; ++k) { for my share of labels */
          make_window(objects[i], labels[j], labels[k], win_ijk);
          h = 0; found = 0;
          while ((h < number_of_objects) && (!found)) {
            if (p[k][h] && in_window(objects[h], win_ijk)
              found = compatible(objects[i], labels[j], objects[h], labels[k]);
            ++h;
            } /* while ((h < number_of_objects) ... */
          if (found)
            ++card_s;
          } /* for (k = ... */

        RECEIVE CONTRIBUTIONS FROM CHILDREN (REDUCTION).

        card_s += left_child_contribution;
        card_s += right_child_contribution;

        if (card_s < q) {
          flag = 1;
          p[j][i] = 0;
          } /* if (card_s ... */

        SEND CHANGES TO THE PE WITH PAIR (i,j).

        } /* if (p[j][i] ... */
  } /* while (flag) ... */
```

Figure 8: Client code for parallel image matching algorithm.

implemented on a serial machine. The programming language was 'C'[Kernigan and Ritchie, 1978]. In figures 8 and 9 we show program segments for the client and server processes, respectively, also written in 'C'. Note that the algorithm-specific constructs are identical in the serial and parallel programs. The only differences are the inclusion of the subroutine calls to perform inter-processor communication. Therefore, one can conclude that the complexity of the parallel software is no greater than that of the serial implementation. This is attributable to the fact that the parallel implementation was designed based on the structure of the algorithm, and not on the structure of the parallel processor architecture.

Programmer burden is a measure of the degree of difficulty in developing and maintaining the parallel algorithm implementation. It can also be viewed as the amount of effort (cost) required to modify and debug the parallel software in light of algorithm modifications. In computer vision, this measure is critical due to the fact that the vision problem is far from being solved and algorithm refinements arrive at a high rate.

As discussed above, the software for the parallel implementation of the image matching algorithm is identical to that of the serial implementation as far as algorithm specific constructs are concerned. Therefore, algorithm debugging and modification can take place in the serial environment where advanced tools are readily available and then ported directly to the parallel environment. Using this technique, once we achieved a "bug free" version of the algorithm on a serial computer, we were able to

```
done = 0;
while (!done) {

  RECEIVE OBJECT/LABEL PAIR FROM PARENT (BROADCAST).

  SEND OBJECT/LABEL PAIR TO CHILDREN (BROADCAST).

  /* Compute the degree of support for the object/label */
  /* assignment provided by this PEs 1/nth of the label table. */

  contribution = 0;
  for (k = 0; k < my_share; ++k) { for my share of labels */
    make_window(objects_i, labels_j, labels[k], win_ijk);
    k = 0; found = 0;
    while ((k < number_of_objects) && (!found)) {
      if (p[k][k] && in_window(objects[k], win_ijk)
        found = compatible(objects[i], labels[j], objects[k], labels[k]);
      ++k;
      } /* while ((k < number_of_objects) ... */
    if (found)
      ++card_s;
      } /* for (k = ... */

    RECEIVE CONTRIBUTIONS FROM CHILDREN (REDUCTION).

    card_s += left_child_contribution;
    card_s += right_child_contribution;

    REPORT THE CONTRIBUTION TO THE PARENT (REDUCTION).

    RECEIVE CHANGES WHEN APPLICABLE.

    SEND CHANGES TO CHILDREN WHEN APPLICABLE.

    } /* while (flag) ... */
```

Figure 9: Server code for parallel image matching algorithm.

get it running in parallel in approximately twelve hours. The primary effort was in validating the inter-process communication. But, once validated, the code that implements the communication is functionally portable to other algorithms that utilize the same communication network topology and need not be validated again.

Therefore, we can conclude that we have minimized the measure of programmer burden in that the development and maintenance efforts for the parallel implementation were performed, predominately, in the serial environment.

This image matching algorithm was previously mapped onto a parallel processor architecture via the classical approach of specifying an architecture then mapping the algorithm onto it [Reisis and Prasanna-Kumar, 1987]. In that study the specified architecture was a *2D mesh* connected SIMD architecture consisting of relatively simple processing elements, a parallel processor architecture well suited to low-level computer vision tasks. By using elaborate data partitioning and memory access schemes, an implementation that theoretically achieves linear speedup (in the absence of data dependencies) was developed. When data dependencies are considered, the implementation achieves the same measures of algorithm speedup and processor efficiency as we presented. The study was purely theoretical and implementation was not actually carried out. To actually implement the system would be extremely difficult due to the nature of the data partitioning scheme, which is retinotopic based to match the communication network topology of the architecture. Furthermore, one can show that the effects of the data dependencies on speedup and efficiency are exaggerated due to synchronous nature of the SIMD machine. Finally, any modification of the implementation (algorithm) would require intimate knowledge of the algorithm, the architecture, and the implementation as is the case with most "classical" parallel algorithm implementations.

We have shown that these situations can be overcome by design (or selection) of a parallel processor architecture based on the processing and data requirements of the algorithm rather than specifying the algorithm implementation to meet the specifications of the parallel processor architecture.

4 Summary

We have described a methodology for mapping algorithms onto parallel processor architectures. Utilizing our methodology to analyze, simulate, and implement a commonly used algorithmic technique, relaxation, we exposed various characteristics common among high-level vision algorithms that must be considered when designing a parallel implementation if the goals of maximized algorithm speedup, maximized processor efficiency, minimized system complexity, and minimized programmer burden are to be achieved. These characteristics include: 1) the use of complex program logic; 2) the existence of subtle data dependencies; 3) the use of heterogeneous data structures; and 4) the dynamic nature of the data.

As shown in [Reisis and Prasanna-Kumar, 1987], when designing an implementation targeted for a specific parallel processor architecture these issues either cannot be sufficiently addressed or require extremely convoluted, unintuitive solutions which lead to an implementation which is difficult to develop and maintain. In applying our methodology we have shown that all of these issues can be addressed without sacrificing any of the goals.

We are currently applying the methodology to other stand-alone computer vision algorithms as well as to complete computer vision systems to determine its utility in specifying a reconfigurable or heterogeneous parallel processor architecture for implementation of such an algorithm suite. We are also investigating the usefulness (cost versus payoff) of dynamic data partitioning (load balancing) schemes in there application to high-level vision algorithm implementations.

References

[Ahuja and Swamy, 1984] N. Ahuja and S. Swamy. Multiprocessor pyramid architectures for bottom-up image analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(4):463–475, July 1984.

[Bokhari, 1981] S. H. Bokhari. On the mapping problem. *IEEE Transactions on Computers*, C-30(3):207–215, March 1981.

[Cole and Vishkin, 1986] R. Cole and U. Vishkin. Approximate and exact parallel scheduling with applications to list tree and graph problems. In *Proceedings of the 27th Symposium on Foundations of Computer Science*, pages 478–491, 1986.

[Faugeras and Price, 1981] O.D. Faugeras and K. Price. Semantic description of aerial images using stochastic labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-3(6):633–642, November 1981.

[Flynn, 1972] M. J. Flynn. Some computer organizations and their effectiveness. *IEEE Transactions on Computers*, C-21(9):948–960, September 1972.

[Ghahraman *et al.*, 1980] D. E. Ghahraman, A. K. C. Wong, and T. Au. Graph monomorphism algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-10(4):189–196, April 1980.

[Hamey *et al.*, 1988] L. G. C. Hamey, J. A. Webb, and I-C. Wu. Low-level vision on Warp and the Apply programming model. In J. S. Kowalik, editor, *Parallel Computation and Computers for Artificial Intelligence*, pages 185–199. Kluwer Academic Publishers, 1988.

[Hwang and Briggs, 1984] K. Hwang and F. A. Briggs. *Computer Architecture and Parallel Processing*. McGraw-Hill, New York, 1984.

[INM, 1989] INMOS Corporation. *The Transputer Databook*, 1989.

[Kernigan and Ritchie, 1978] B. W. Kernigan and D. M. Ritchie. *The C Programming Language*. Prentice Hall, New Jersey, 1978.

[Kuehn *et al.*, 1985] J. T. Kuehn, H. J. Siegel, D. L. Tuomenoksa, and G. B. Adams III. The use and design of PASM. In S. Levialdi, editor, *Integrated Technology for Parallel Image Processing*, pages 133–152. Academic Press, 1985.

[Levitan, 1984] S. P. Levitan. *Parallel Algorithms and Architectures: A Programmer's Perspective*. PhD thesis, University of Massachusetts, 1984. Computer and Information Sciences.

[Little *et al.*, 1987] J. J. Little, G. Blelloch, and T. Cass. Parallel algorithms for computer vision on the Connection Machine. In *Proceedings of the DARPA Image Understanding Workshop*, pages 628–638, February 1987.

[Medioni and Nevatia, 1984] G. Medioni and R. Nevatia. Matching images using linear features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):675–685, November 1984.

[Reisis and Prasanna-Kumar, 1987] D. Reisis and V. K. Prasanna-Kumar. Parallel processing of image and stereo matching using linear segments. Technical Report IRIS 204, University of Southern California, February 1987.

[Rice and Jamieson, 1985] T. A. Rice and H. Jamieson. Parallel processing for computer vision. In S. Levialdi, editor, *Integrated Technology for Parallel Image Processing*, pages 57–78. Academic Press, 1985.

[Rosenfeld and Kak, 1976] A. Rosenfeld and A. Kak. *Digital Image Processing*. Academic Press, New York, 1976.

[Rosenfeld and Smith, 1981] A. Rosenfeld and R. C. Smith. Thresholding using relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-3(5):598–606, September 1981.

[Rosenfeld *et al.*, 1976] A. Rosenfeld, R. A. Hummel, and S. W. Zucker. Scene labeling by relaxation operations. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-6(6):420–433, June 1976.

[Rosenfeld *et al.*, 1986] A. Rosenfeld, B. Simpson, and Squires S., editors. *Proceedings of the DARPA Workshop on Architectures for Image Understanding*, McLean, Virginia, November 1986.

[Rutkowski *et al.*, 1981] W. S. Rutkowski, S. Peleg, and A. Rosenfeld. Shape segmentation using relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-3(4):368–375, July 1981.

[Stout, 1988] Q. F. Stout. Mapping vision algorithms to parallel architectures. *Proceedings of the IEEE*, 76(8):982–995, August 1988.

[Sunwoo and Aggarwal, 1989] M. H. Sunwoo and J. K. Aggarwal. VisTA: An integrated vision tri-architecture system. Technical Report CVRC TR-89-6-57, University of Texas at Austin, September 1989. Laboratory for Image and Signal Analysis.

[Terzopoulos, 1986] D. Terzopoulos. Image analysis using multigrid relaxation methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(2):129–139, March 1986.

[Waltz, 1972] D. Waltz. *Generating Semantic Descriptions from Drawings of Scenes with Shadows*. PhD thesis, Massachusetts Institute of Technology, November 1972. Artificial Intelligence Laboratory.

[Weems and Levitan, 1987] C. C. Weems and S. P. Levitan. The Image Understanding Architecture. In *Proceedings of the DARPA Image Understanding Workshop*, pages 483–496, February 1987.

[Weems, 1988] C. C. Weems. Some sample algorithms for the image understanding architecture. In *Proceedings of the DARPA Image Understanding Workshop*, pages 127–128, April 1988.

B-SPLINE CONTOUR REPRESENTATION AND SYMMETRY DETECTION[1]

Philippe Saint-Marc and Gérard Medioni

Institute for Robotics and Intelligent Systems
University of Southern California
Los Angeles, California 90089-0273

October 20, 1989

[1]This research was supported by the Defense Advanced Research Projects Agency under contract F33615-87-C-1436 monitored by the Wright-Patterson Air Force Base.

Abstract

The detection of edges is only one of the first steps in the understanding of images. Further processing necessarily involves grouping operations between contours. We present a representation of edge contours by approximating B-splines and show that such a representation facilitates the extraction of symmetries between contours. Our representation is rich, compact, stable, and does not critically depend on feature extraction whereas interpolating splines do. We turn our attention to the detection of two types of symmetries, skewed and parallel, which have proven to be of great importance to infer shape from contour, and show that our representation is computationally attractive. As an application, we show how parallel symmetries can be used to infer the 3-D orientation of a torus from its intensity image.

1 Introduction

Edge detection is not a goal in itself, but has to be considered as one of the steps in the processes involved to understand images. The question therefore arises of representing the contours formed by edgels (edge elements).

Iconic representations, such as edge maps, or chain codes [11] do not make the necessary information explicit: by definition edgels only capture very local properties of an image, and the inference of higher structures, such as object boundaries, requires *grouping* operations. We believe that such operations rely on basic and simple properties and various forms of symmetry [18]. The representation must therefore make explicit differential properties of contours, such as tangent and curvature. Furthermore, because of the variability inherent

in the imaging process, the representation should be tolerant to noise, partial occlusion, and perspective, naturally suggesting segmented, local descriptors [30].

If the world was composed of polyhedral objects alone, we would know to expect only straight line segments in images, and polygonal approximations such as performed by the Linear package [22] or Hough transforms [2] would be appropriate. In many cases, such approximation is indeed appropriate, as demonstrated by several applications such as stereo [16], aerial image understanding [13] or object recognition [19, 33], but is unable to capture curvature information, since it is a first order approximation. Also, if a contour is smooth, the number of points required to approximate it may be quite large, and the exact position of the points somewhat unrelated to the contour itself. Another possibility is to use a mixture of curves and lines [25] but it leads to unstabilities of the description when we switch from one to another.

These issues have been tackled by the graphics community in the context of design, and we propose to use some of the resulting tools, particularly B-splines.

In the next section, we briefly review some of the successful applications of B-splines to contour representation in computer vision, but propose that approximating splines are more appropriate than interpolating splines because they are more tolerant of segmentation errors. We derive the equations and present results demonstrating that the resulting representation is compact and faithful to the original data for smooth or piecewise smooth contours, open or closed.

We then turn our attention to an application ideally suited for our representation, the detection of symmetries. Whereas it is easy to define symmetry between two infinite straight lines, the concept of symmetry between curves is harder to define: Rosenfeld [31]

provides a lucid account of the differences between Blum's [4], Brooks' [7], and Brady's [5] definitions, and a more recent paper by Ponce [28] gives further comparisons. Here, we are interested not in local symmetries which provide skeletal shape primitives, but rather in symmetries which help to infer shape from contour: Nevatia and Ulupinar [34] postulate that they are skewed and parallel.

We recall these definitions in section 3, and show how each one can be extracted using our B-spline representation. The most obvious advantages are the low computational complexity of the process and the stability of the results. Finally, we show that for the very specific case of a torus, the detection of parallel symmetries allows to infer the 3-D orientation of the object in a much simpler fashion than proposed in [29].

2 Contour Representation

A very promising idea for representing image contours is to use piecewise polynomials. The advantages are obvious: this representation is rich, compact, analytical and local in the sense that a small change in the original curve does not affect the representation entirely.

The approach commonly used consists of first extracting a set of knots from the discrete curve and then to approximate the curve between each pair of knots by polynomials under continuity constraints at the knots. In [27], the knots are extracted by taking the vertices of a polygonal approximation, then a finite search technique such as dynamic programming is used to select those knots which provide the best approximation by cubics. A slightly different idea is proposed in [21] where the initial knots are selected by taking the vertices of the polygonal approximation of the tangent orientation signal $\theta(s)$. The final knots are

determined using a split and merge algorithm. Conics are used instead of cubics. In [15], the initial knots are the zero-crossings and extrema of the curvature computed after different amounts of Gaussian smoothing and are then selected using again dynamic programming. Monotone curvature splines are used to interpolate through the selected knots. In [1], the knots are the corners and smooth joins marked in the *curvature primal sketch*. Results using circular splines are shown. Finally, an elegant algorithm is proposed in [17] which *at the same time* locates corners and encodes curve segments between them using cubic B-splines.

The main point that we formulate against these methods is that they are too much based on the always critical segmentation step which brings up the stability issue. Also techniques such as dynamic programming can yield a complexity of $O(n^3)$ where n is the number of initial knots [27]. Finally, a simple case such as a circle, from which no curvature features can be extracted, is a typical example of a curve which on the other hand could easily be approximated using the following B-spline least-squares fitting method which, as we will see, does not require any knot selection and is relatively insensitive to noise.

2.1 B-spline Least-Squares Curve Fitting

Although there are numerous textbooks on the subject of B-splines (see [3, 10] for example), it is useful to recall some basic definitions and properties. It is well known that a B-spline is a piecewise polynomial. Cubic polynomials are often used since they are the lowest order for which the curvature can change sign. A B-spline is expressed as a linear combination of basis functions which are themselves piecewise polynomials, the coefficients being the vertices of the B-spline *guiding polygon*. Thus, a B-spline can be easily manipulated by

modifying its guiding polygon, hence its popularity in CAD systems. Furthermore, as B-splines are defined locally, modifying the position of a vertex does not affect the B-spline entirely. In the case of a planar curve, a B-spline $Q(u) = (X(u), Y(u))$ with $m+1$ vertices is defined as follows:

$$Q(u) = \sum_{j=0}^{m} V_j B_j(u) = \sum_{j=0}^{m} (X_j B_j(u), Y_j B_j(u))$$

In the above equations, the (X_j, Y_j) are the *vertices* of the guiding polygon and $B_j(u)$ the basis functions.

Let C be an ordered set of $p+1$ points $P_i = (x_i, y_i)$, what is the B-spline which best approximates C? An approach proposed in [3] consists of minimizing the distance

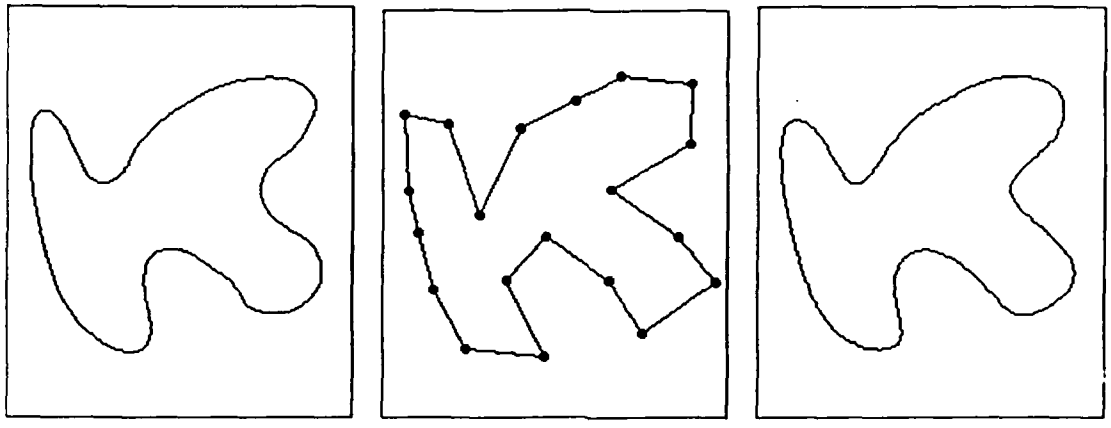
$$\begin{aligned} R &= \sum_{i=0}^{p} \|Q(u_i) - P_i\|^2 \\ &= \sum_{i=0}^{p} (\sum_{j=0}^{m} X_j B_j(u_i) - x_i)^2 + (\sum_{j=0}^{m} Y_j B_j(u_i) - y_i)^2 \end{aligned}$$

In the above formulation, u_i is some parameter value associated with the i^{th} data point [3]. Minimizing R is equivalent to setting all partial derivatives $\partial R/\partial X_l$ and $\partial R/\partial Y_l$ to 0, for $0 \leq l \leq m$, which yields

$$\sum_{j=0}^{m} X_j \sum_{i=0}^{p} B_j(u_i) B_l(u_i) = \sum_{i=0}^{p} x_i B_l(u_i) \quad (1)$$

$$\sum_{j=0}^{m} Y_j \sum_{i=0}^{p} B_j(u_i) B_l(u_i) = \sum_{i=0}^{p} y_i B_l(u_i) \quad (2)$$

$$\text{with } 0 \leq l \leq m$$

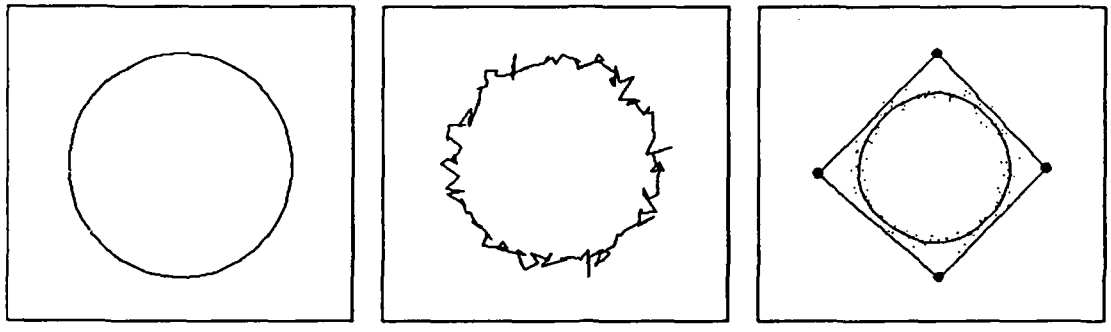


(a) Original Curve (b) Guiding Polygon (c) Reconstruction

Figure 1: Example of Planar Curve Cubic B-Spline Fit

The linear systems (1) and (2) are easily solved for all X_j and Y_j respectively using standard linear algebra, yielding the guiding polygon of the B-spline which best approximates the original curve. In the case of open curves, we have the option to force end-points to be interpolated. In this case, the first and last vertices are simply set to lie at the end-points so that the linear systems are reduced to $m-1$ equations of $m-1$ unknowns. In the case of closed curves, the linear systems are over constrained since some vertices are required to be identical. Hence the pseudo-inverse method [2] can be used. For more details about B-splines and end conditions, see [3].

As an example, figure 1(a) shows a free-form planar curve, figure 1(b) the guiding polygon of the approximating cubic B-spline, and figure 1(c) the reconstructed curve from the guiding polygon. Note that not only the reconstructed curve is almost identical to the original curve, but the internal representation of the approximating curve consists of only a small number of vertices (20 in this example).



(a) Original Curve

(b) Gaussian Noise Added

(c) B-Spline and Guiding Polygon overlayed on Noisy Data

Figure 2: A Cubic Spline Fit on Noisy Data

Another example displayed in figure 2 illustrates the fact that this method is relatively insensitive to noise, hence stable. Figure 2(a) shows a circle and figure 2(b) the resulting curve after having added Gaussian noise to the data points. Figure 2(c) shows the result of fitting a cubic B-spline to the noisy data using 4 vertices.

The choice of m (the number of vertices) determines how close to the original data the approximation is, which is measured by R (see above). The automatic selection of the number of vertices is not trivial. Our approach is to premilinary set a fitting tolerance r_0 and find the value of m which yields the normalized distance $r = R/(p+1)$ closer to r_0 using a binary search approach.

2.2 From Edgels to B-Splines

The input for our system is an edge map produced by an edge detector such as Canny's [9]. Three stages are sequentially considered: linking, corner detection, and B-spline approximation.

2.2.1 Linking

Although there exists numerous linking methods proposed in the litterature [2], we use a simple and fast scheme which can be summarized as follows. No gap-bridging or other task is performed, the goal being a fast extraction of the elementary curves present in the image. It is also our belief that point-wise surgery is too myopic, and that if grouping is needed, it has to be performed at a higher level [18]. The condition for the algorithm to work correctly is that the input edges have to be 8-connected. If it is not the case, then a simple [8] or a more elaborate [24] thinning algorithm has to be applied before going further. The first stage consists of labelling edge points according to the number of neighboors they have in order to distinguish among three types of edgels: *end-points, junctions, and mid-points.* Then an edge follower is applied starting from end-points and stopped when other end-points or junctions are encountered. The resulting open curves are stored and the corresponding points deleted from the edge map except for the junctions. The same procedure is applied but starting from junctions and stopped when other junctions are met. The only remaining curves in the edge map should be closed and are finally extracted by applying the edge follower starting from any mid-point.

2.2.2 Corner Detection

The detection of corners is essential for the description of planar curves. Corners correspond to tangent discontinuities to which human beings are very sensitive.

We have recently developped an algorithm called *adaptive smoothing* [32] based on the *anisotropic diffusion* principle [26], which consists of smoothing a signal while preserving

and even enhancing its significant discontinuitues. This is achieved by repeatedly convolving the signal with a very small averaging filter modulated by a measure of the signal discontinuity at each point. The method is extremely attractive since a single parameter k fixes the amplitude of the discontinuities to preserved. Those features are then easy to detect and directly localized. Hence no coarse to fine correspondence problem has to be solved as it is the case in the *curvature primal sketch* [1].

When the signal consists of the tangent orientation $\theta(s) = \arctan \frac{y'(s)}{x'(s)}$ of a planar curve $C(s) = (x(s), y(s))$, the adaptive smoothing process tends to preserve and enhance tangent orientation discontinuities which correspond to corners. Corner detection then consists in differentiating the smoothed version of $\theta(s)$ and extract the local extrema of the resulting signal which lie above a threshold which we set equal to the smoothing parameter k.

2.2.3 B-spline Approximation

Once the image edge points have been linked into curves and the corners detected, the final step toward image contours representation consists of approximating by a B-spline each elementary curve. When a closed curve with no corners is considered, a global least-squares approximation is performed. In the case of an open curve or a closed curve with corners, each curve segment between pairs of corners is approximated with the constraint that the end-points have to be interpolated. This insures the reconstructed curves to be continuous at the corner locations.

Figure 3 summarizes the complete process on an example. The 167×222 intensity image of a Mozart bust is displayed in figure 3(a) and the contour points obtained after applying a Canny type edge detector in figure 3(b). Those edge points have been linked into

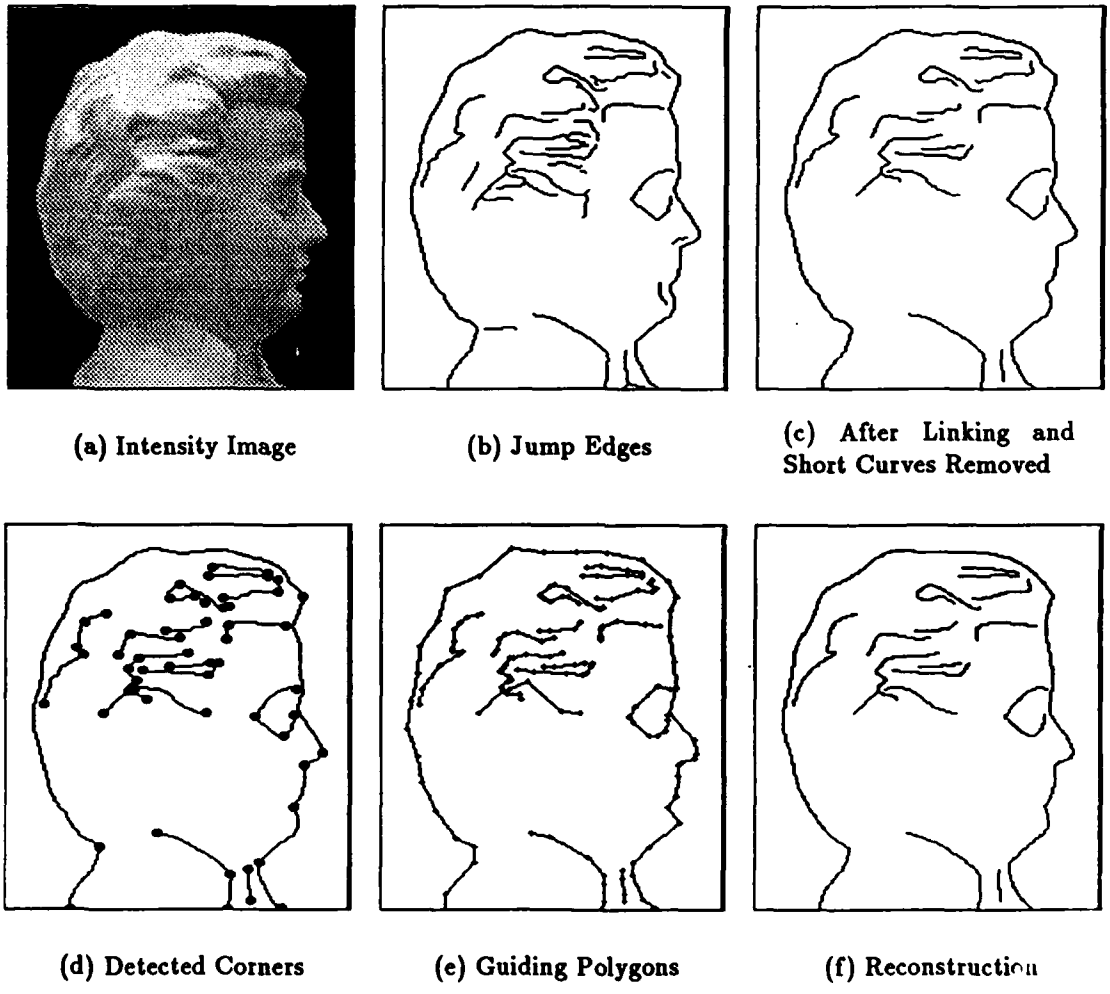
elementary curves and figure 3(c) shows the longest curve found. The corners detected after adaptive smoothing are displayed in figure 3(d), the end-points being marked as corners as well. Finally, a quadratic B-spline approximation of each curve segment between corners has been done using a fitting tolerance of 0.5 which led to the guiding polygon displayed in figure 3(e) and the corresponding quadratic B-spline displayed in figure 3(f).

Finally, in order to illustrate that our method is very tolerant to segmentation errors, we show some results obtained after different corner detections. Figure 4(a) shows the result of a first corner detection on the contour of a telephone handset, and figure 4(b) and figure 4(c) show the guiding polygon and the reconstructed curve obtained using a quadratic B-spline approximation. If a corner is missed, as shown in figure 4(d), then the guiding polygon shown in figure 4(e) is obtained. The reconstructed curve displayed in figure 4(f) is very similar to the one obtained with the additional corner.

In the following section, we show how piecewise polynomial representations of image contours can be used for detecting symmetries in the image plane.

3 Symmetry Detection

The detection of symmetries is an essential step when inferring shapes from contours [35]. It has been shown [14] that skewed symmetries can be used for recovering the 3-D structure of polyhedra from their 2-D line drawings. In the case of the surfaces of revolution [20], the orthographic projection of their limbs exhibits reflectional symmetry, the axis of revolution being the back-projection of the symmetry axis. More recently [34], two kinds of symmetries have been proposed to give significant information about the surface shape for a variety of



(a) Intensity Image

(b) Jump Edges

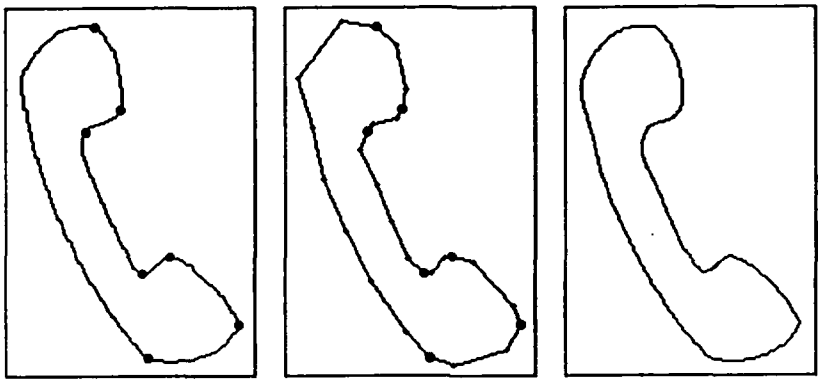
(c) After Linking and Short Curves Removed

(d) Detected Corners

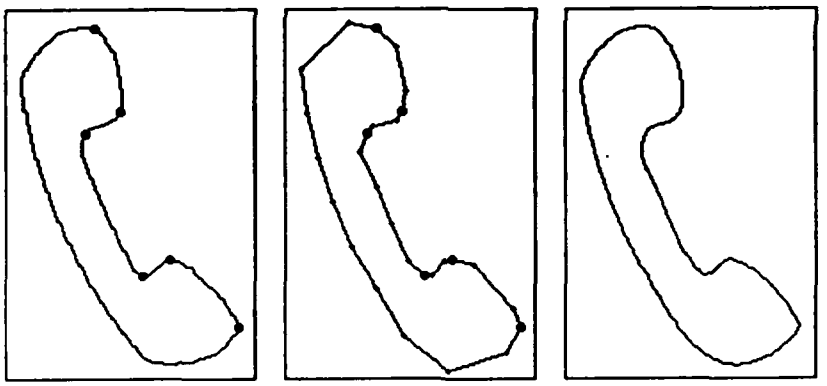
(e) Guiding Polygons

(f) Reconstruction

Figure 3: Example of the Overall Process on a Mozart Bust Image



(a) Detected Corners (b) Guiding Polygon (c) Reconstruction



(d) Detected Corners (e) Guiding Polygon (f) Reconstruction

Figure 4: If a Corner is Missed...

3-D objects: skewed and parallel symmetries. The method is applicable to Zero Gaussian Curvature surfaces, and to a variety of doubly curved surfaces. Results on the recovery of surface orientation for cylindrical and conic objects are shown but the method still need to be validated on real data. We propose to use the contour representation previously exposed for detecting skewed and parallel symmetries. In the remainder of this paper, the lines joining symmetric points will be called *lines of symmetry* and the mid-points of these lines will form the *axis of symmetry*.

3.1 Skewed Symmetry

3.1.1 Previous Work

The detection of skewed symmetries has been investigated by several researchers for the past few years. [12, 29, 35]. In [12], the method is based on the moments of a figure, hence limiting the detection to non-occluded objects. In [29], a local approach, like for the detection of smooth local symmetries [5], is used. It consists of using a local property of the skewed symmetry in order to identify symmetric edge points. It is thus necessary to test every possible pair of edge points against the property which leads to an $O(n^2)$ algorithm where n is the number of points. However, it is possible to reduce the complexity by using the method of projections [23]. Finally a method using the Hough transform is proposed in [35] but as opposed to the two previous papers, no results are shown.

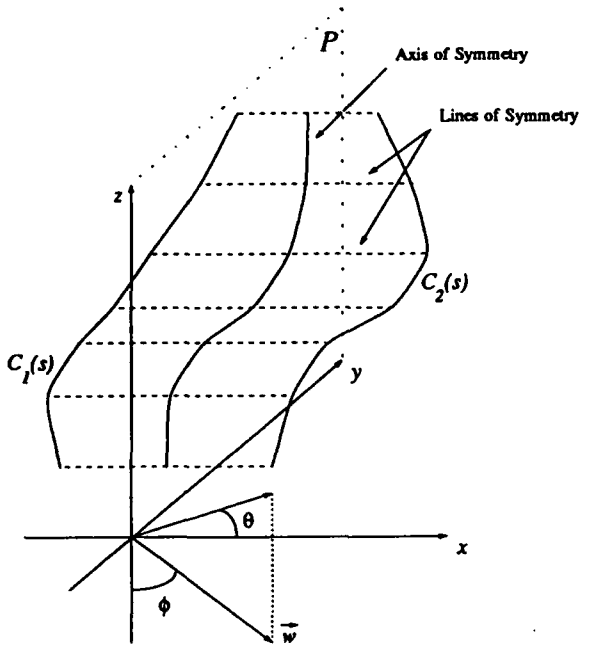


Figure 5: 3-D Mirror Symmetry

3.1.2 Theoretical Study

Let us suppose that we are given two parametric 3-D curves $C_1(s)$ and $C_2(s)$ which are 3-D mirror symmetric with respect to a plane $\mathcal{P}$ (see figure 5). Without loss of generality, we can choose a coordinate system such that we have

$$C_1(s) = \begin{pmatrix} x(s) \\ y(s) \\ z(s) \end{pmatrix} \quad \text{and} \quad C_2(s) = \begin{pmatrix} -x(s) \\ y(s) \\ z(s) \end{pmatrix}$$

If we project these curves along $\vec{w}$ onto the image plane $\mathcal{I}(\vec{u}, \vec{v})$ such that

$$\vec{w} = \begin{pmatrix} \cos(\theta)\sin(\phi) \\ \sin(\theta)\sin(\phi) \\ -\cos(\phi) \end{pmatrix} \quad \vec{u} = \begin{pmatrix} \cos(\theta)\cos(\phi) \\ \sin(\theta)\cos(\phi) \\ \sin(\phi) \end{pmatrix} \quad \vec{v} = \begin{pmatrix} \sin(\theta) \\ -\cos(\theta) \\ 0 \end{pmatrix}$$

we obtain the parametric planar curves $c_1(s) = (u_1(s), v_1(s))$ and $c_2(s) = (u_2(s), v_2(s))$ where

$$\begin{cases} u_1(s) &= \cos(\theta)\cos(\phi)x(s) + \sin(\theta)\cos(\phi)y(s) + \sin(\phi)z(s) \\ v_1(s) &= \sin(\theta)x(s) - \cos(\theta)y(s) \end{cases}$$

and

$$\begin{cases} u_2(s) &= -\cos(\theta)\cos(\phi)x(s) + \sin(\theta)\cos(\phi)y(s) + \sin(\phi)z(s) \\ v_2(s) &= -\sin(\theta)x(s) - \cos(\theta)y(s) \end{cases}$$

The curve $a(s) = (u_a(s), v_a(s))$ such that

$$\begin{cases} u_a(s) = \frac{u_1(s)+u_2(s)}{2} &= \sin(\theta)\cos(\phi)y(s) + \sin(\phi)z(s) \\ v_a(s) = \frac{v_1(s)+v_2(s)}{2} &= -\cos(\theta)y(s) \end{cases}$$

is simply the projection of the curve $(0, y(s), z(s))$ which is the axis of the 3-D mirror symmetry with respect to the plane $\mathcal{P}$. It is easy to verify that $\frac{v_2(s)-v_1(s)}{u_2(s)-u_1(s)} = \frac{\tan(\theta)}{\cos(\phi)} = a_0$, hence the lines of symmetry are parallel to each other in the image plane and have the same direction $\alpha_0 = \arctan(a_0)$. If $C_1(s)$ and $C_2(s)$ are co-planar, i.e. $y(s)$ and $z(s)$ are linear, we have the following well-known result that $a(s)$ has to be straight, hence $c_1(s)$ and $c_2(s)$ are skewed symmetric with respect to $a(s)$.

Now given two curves $c_1(s)$ and $c_2(s)$ in the image plane, is it possible to eventually identify a unique 3-D mirror symmetry. Let us take the example of figure 6. Figure 6(a)

shows the contour of a symmetric planar object viewed from an arbitrary direction and figure 6(b) a window taken out of the previous image. If we choose a direction α_0 and draw the line segments joining each point of a curve with the corresponding point in the other curve along that direction, these segments define the lines of symmetry of some 3-D mirror symmetry. Figure 6(d) shows what happens when $\alpha_0 = 90°$ and figure 6(c) the corresponding axis of symmetry. Figure 6(e) and 6(f) show another case when $\alpha_0 = 160°$. This example illustrates that in fact any value for α_0 is a possible answer.

If however we are looking for 3-D mirror symmetries between *planar* curves, the answer is clear since we saw that the axis of symmetry has to be straight. In this case, the problem consists of finding a value for θ_0 which yields a straight axis. Figure 6(g) and 6(h) show the results obtained when the "correct" value for θ_0 has been found. The following describes how the quadratic B-spline representation of image contours helps us in finding skewed symmetries.

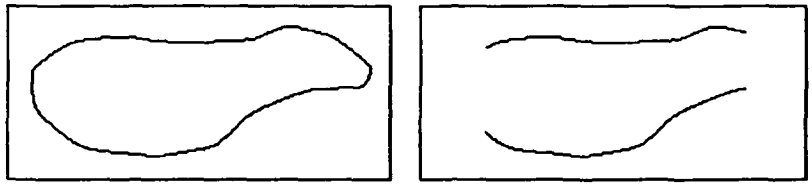
3.1.3 Quadratic B-Spline Implementation

Let $c_1(u) = (x_1(u), y_1(u))$ and $c_2(v) = (x_2(v), y_2(v))$ be two conic segments (see figure 7), both defined over the interval $[0, 1]$, given by the equations

$$\begin{cases} x_1(u) &= a_{x_1}u^2 + b_{x_1}u + c_{x_1} \\ y_1(u) &= a_{y_1}u^2 + b_{y_1}u + c_{y_1} \end{cases}$$

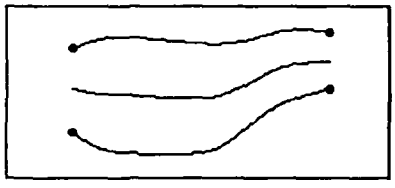
and

$$\begin{cases} x_2(v) &= a_{x_2}v^2 + b_{x_2}v + c_{x_2} \\ y_2(v) &= a_{y_2}v^2 + b_{y_2}v + c_{y_2} \end{cases}$$

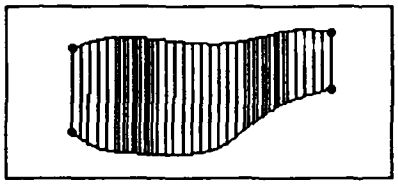


(a) Image Contour

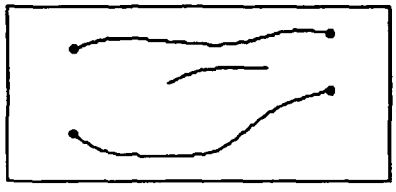
(b) Portion of (a)



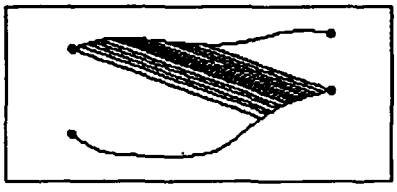
(c) Symmetry Axis (90°)



(d) Lines of Symmetry

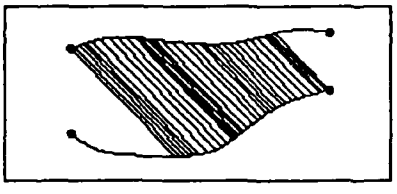


(e) Symmetry Axis (160°)



(f) Lines of Symmetry

(g) Straight Axis



(h) Lines of Symmetry

Figure 6: A Case Study on the Contour of a Planar Object

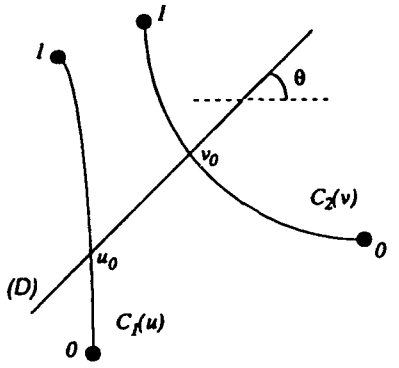


Figure 7: Two Conic Segments

Given a direction θ, let $Ax + By + C = 0$ be the implicit equation of a straight line $(\mathcal{D})$ along that direction. Suppose that $(\mathcal{D})$ intersects $c_1(u)$ at $u_0 \in [0,1]$, what is the corresponding value(s) of v for which $(\mathcal{D})$ intersects $c_2(v)$? After some straightforward manipulations, it comes out that v is the root of the quadratic equation

$$\mathcal{A}v^2 + \mathcal{B}v + \mathcal{C} = 0$$

where

$$\begin{aligned}
\mathcal{A} &= Aa_{x_2} + Ba_{y_2} \\
\mathcal{B} &= Ab_{x_2} + Bb_{y_2} \\
\mathcal{C} &= Ac_{x_2} + Bc_{y_2} - (Aa_{x_1} + Ba_{y_1})u_0^2 - (Ab_{x_1} + Bb_{y_1})u_0 - (Ac_{x_1} + Bc_{y_1})
\end{aligned}$$

Hence, given two conic segments and a direction θ, it is relatively simple to establish the mapping from one segment to the other along that direction. The mid-points of the lines of symmetry found form the axis of the 3-D mirror symmetry between the two conic segments. Note that in many cases, there might not be solutions such that $v_0 \in [0,1]$ (the details are skipped for simplicity purpose). As a quadratic B-spline can be expressed as a collection of

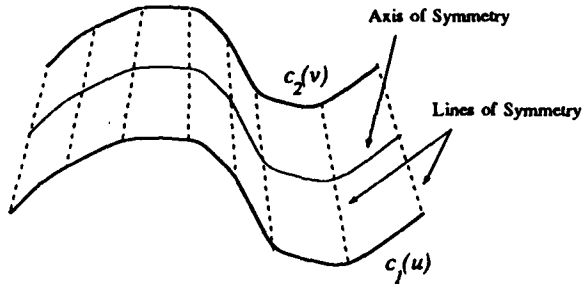


Figure 8: Parallel Symmetry

connected conic segments $\mathcal{S} = \{c_i(u)\}$, for $i = 0, \cdots, m$, each defined on the interval $[0, 1]$. Given another quadratic B-spline $\mathcal{S}' = \{c'_j(v)\}$, for $j = 0, \cdots, n$, each conic segment of $\mathcal{S}$ is compared against each conic segment of $\mathcal{S}'$ to eventually find elementary axis of symmetry. To detect skewed symmetries, we compute the value for θ which minimizes the torsion of the global axis of symmetry obtained after grouping the elementary symmetries. This is achieved by using Brent's minimization technique [6]. Figure 6(g) and 6(h) shows the axis of skewed symmetry found for the contours of figure 6(b). Notice that if the axis found is not straight enough, it is rejected and the conclusion is that there is no skewed symmetry.

3.2 Parallel Symmetry

Let $c_i(s) = (x_i(s), y_i(s))$, for $i = 1, 2$, be two parametric planar curves, and $\theta_i(s)$ their tangent orientation. $c_1(s)$ and $c_2(s)$ are said to be parallel symmetric if there exists a continuous monotonic function $f(s)$ such that $\theta_1(s) = \theta_2(f(s))$. Note that parallel symmetry is an improper term because, mathematically speaking, two curves are parallel symmetric only if $f = \mathrm{Id}$. We use this term to be consistent with [34]. Figure 8 shows an occurence of a parallel symmetry between two curves.

As far as we know, the detection of parallel symmetries has not been investigated so far.

The following describes an attractive method which makes use of the quadratic B-spline representation of image contours.

3.2.1 Quadratic B-Splines Implementation

Suppose that we are given two conics $c_1(u) = (x_1(u), y_1(u))$ and $c_2(v) = (x_2(v), y_2(v))$ where

$$\begin{cases} x_1(u) &= a_{x_1}u^2 + b_{x_1}u + c_{x_1} \\ y_1(u) &= a_{y_1}u^2 + b_{y_1}u + c_{y_1} \end{cases}$$

and

$$\begin{cases} x_2(v) &= a_{x_2}v^2 + b_{x_2}v + c_{x_2} \\ y_2(v) &= a_{y_2}v^2 + b_{y_2}v + c_{y_2} \end{cases}$$

After derivation, we have the parametric equations of their tangent $tan_1(u)$ and $tan_2(v)$ given by

$$tan_1(u) = \frac{2a_{y_1}u + b_{y_1}}{2a_{x_1}u + b_{x_1}} \tag{3}$$

$$tan_2(v) = \frac{2a_{y_2}v + b_{y_2}}{2a_{x_2}v + b_{x_2}} \tag{4}$$

Under which conditions are $c_1(u)$ and $c_2(v)$ parallel symmetric?

Writing $tan_1(u) = tan_2(v)$, the following equations are easily obtained:

$$v = \frac{2(a_{x_1}b_{y_2} - a_{y_1}b_{x_2})u + (b_{x_1}b_{y_2} - b_{y_1}b_{x_2})}{4(a_{y_1}a_{x_2} - a_{x_1}a_{y_2})u + 2(b_{y_1}a_{x_2} - b_{x_1}a_{y_2})} = \frac{Au + B}{Cu + D} = f(u) \tag{5}$$

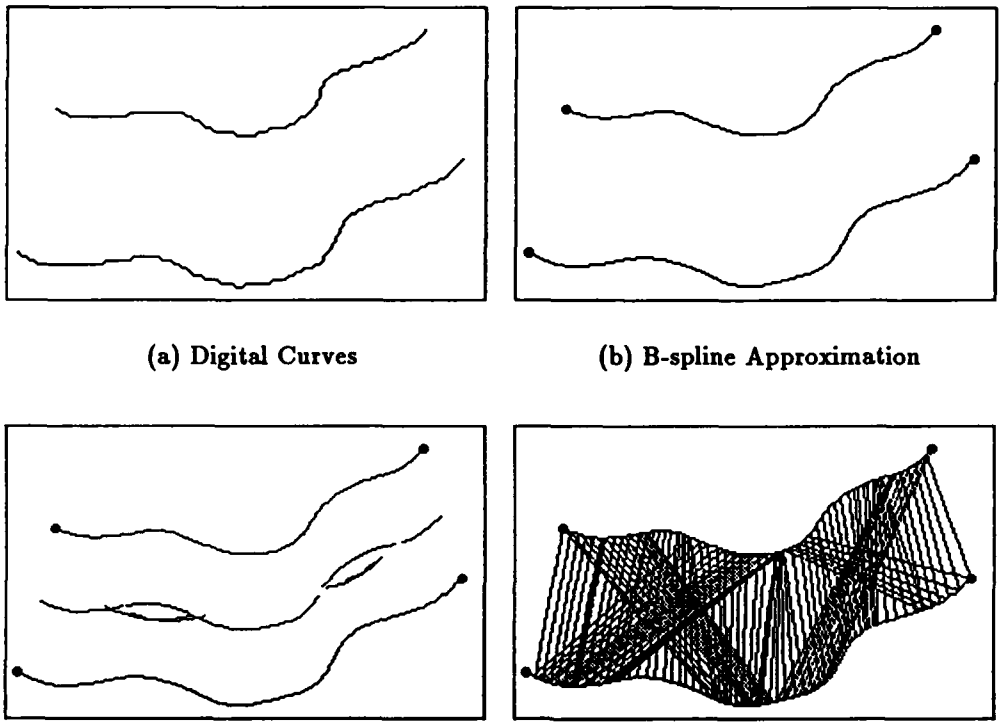
and

$$u = \frac{B - Dv}{Cv - A} = f^{-1}(v) \tag{6}$$

The function $f(u)$ is continuous with a vertical asymptote $u_a = -D/C$ and an horizontal asymptote $v_a = A/C$. Because $f'(u) = \frac{AD-BC}{(Cu+D)^2} \geq 0$, $f(u)$ is monotonic. What happens at u_a and v_a? Substituting u_a into equation (3), we obtain $tan_1(u_a) = \frac{a_{y_2}}{a_{x_2}}$ which equals $tan_2(v)$ when $v \rightarrow \pm\infty$ and substituting v_a into equation (4), we obtain $tan_2(v_a) = \frac{a_{y_1}}{a_{x_1}}$ which equals $tan_1(u)$ when $u \rightarrow \pm\infty$. Hence we have the result that two conics are always parallel symmetric. Now supposing that $c_1(u)$ and $c_2(v)$ are only defined on the interval $[0,1]$, we will say that the two segments $c_1(u)$ and $c_2(v)$ are parallel symmetric on $[u_0, u_1] \subseteq [0,1]$ iff $[f(u_0), f(u_1)] \subseteq [0,1]$ where $f(u)$ is given by equation (5).

Now that we have studied the parallel symmetry between two conic segments, the detection of parallel symmetries between quadratic B-splines is straightforward. A quadratic B-spline can be expressed as a collection of connected conic segments $\mathcal{S} = \{c_i(u)\}$, for $i = 0, \cdots, m$, each defined on the interval $[0,1]$. Note that at the junction between two conic segments, the tangent orientation is continuous. Given another quadratic B-spline $\mathcal{S}' = \{c'_j(v)\}$, for $j = 0, \cdots, n$, each conic segment of $\mathcal{S}$ is compared against each conic segment of $\mathcal{S}'$ to eventually detect an elementary parallel symmetry between them. Given the simplicity of equations (5) and (6), and because of the usually small number of conic segments involved, the method is computationally very efficient.

Figure 9 shows an example of parallel symmetry detection using a quadratic B-spline approximation starting from the two digital curves displayed in figure 9(a). Figure 9(b) shows the quadratic B-spline approximation of the curves of figure 9(a). The detected elementary axis of symmetry are displayed in figure 9(c) while in figure 9(d), the corresponding



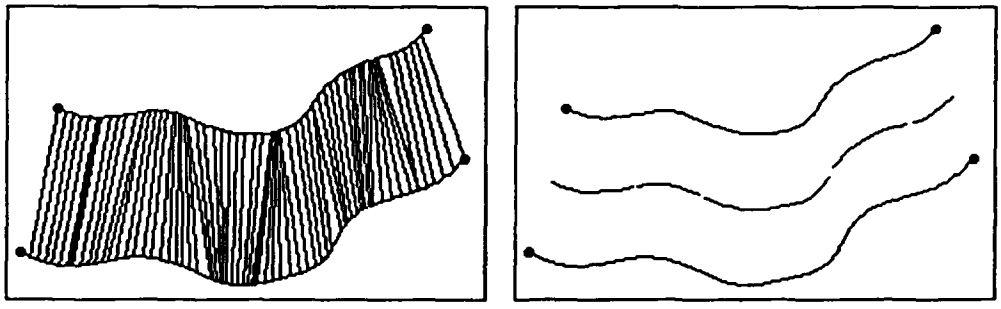
(a) Digital Curves (b) B-spline Approximation

(c) Symmetry Axis (d) Lines of Symmetry

Figure 9: Detection of Elementary Parallel Symmetries

lines of symmetry are shown.

An additional step is needed for selecting those elementary symmetries which can be grouped into a more global expression of the existing symmetries between two curves. In the example of figure 9, it is obvious that some elementary symmetries are purely local whereas some others are part of a more global symmetry. Some grouping is needed and we found that very simple connectivity criteria between elementary symmetries can be used. In figure 10, the largest connected component has been isolated and reflects the global parallel symmetry between the two curves. Strictly speaking, because of the presence of some discontinuities along the axis, the two curves are not parallel symmetric. We used



(a) Lines of Symmetry (b) Symmetry Axis

Figure 10: Selected Global Parallel Symmetry

this example in order to show that once again, a vision task like this one has to deal with noise and imperfections.

3.2.2 The Torus Example

The torus is an interesting example on which to demonstrate the application of parallel symmetry. Assuming that the object is far enough from the camera, and ignoring its actual size, it is reasonable to model the imaging process by an orthographic projection. The torus is a smooth solid of revolution, and the contours generated in its image correspond only to *limbs* or occluding contours, which are unfortunately viewer dependent. The points on these contours are those for which the viewing direction is tangent to the surface.

Ponce and Kriegman [29] have shown that it is possible, although complicated, to express the implicit equation of the contours (a reduced equation still takes 25 lines!), and to use a least-squares method to recover the position and orientation of a torus from its limbs. Instead, we use the property of the torus that the axes of parallel symmetry in its image are the projection of its circular spine (3-D skeletal axis). This property allows

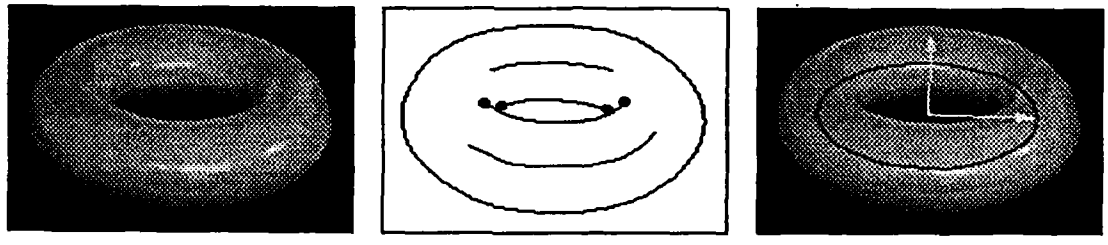
us to recover the 3-D orientation quite simply: we fit an ellipse to the detected parallel symmetry axis, the orientation of the plane on which the torus is lying is given by the eccentricity of the ellipse and the angle of the major axis with the horizontal. Figure 11 shows the results obtained for a torus imaged at two different orientations. The first column shows the intensity images, the second column the contours along with the detected parallel symmetries using the quadratic B-spline representation of the contours, and the last column shows the ellipse fitted to the axis of symmetry overlayed on the intensity image. The two vectors drawn are the projection of two unit vectors in space: the vertical one lying on the axis of revolution, the horizontal one lying in the plane of the torus spine. The entire process, given the contours, takes a few seconds only on a Symbolics machine.

4 Conclusion

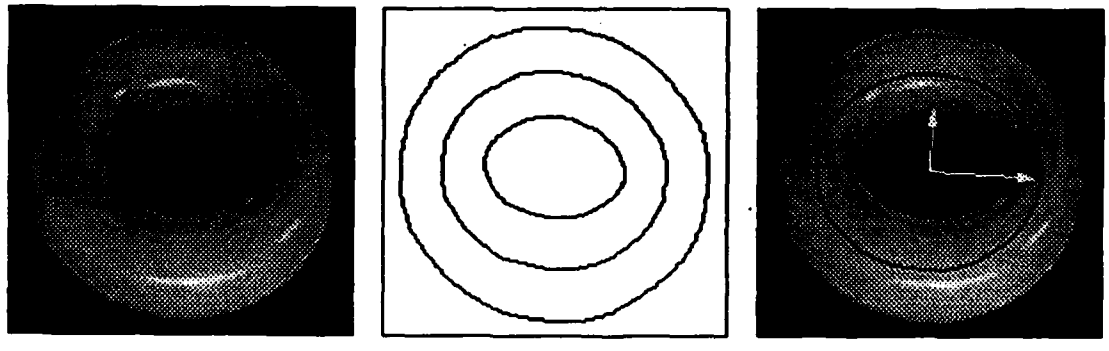
We have presented an approach to representing contours using approximating B-splines. It has attractive properties for use in Computer Vision: the representation is rich, compact, stable, local and segmented. We have shown how this representation can be used to extract two important types of symmetry, skewed and parallel, on contours in real images. We are currently working on the selection of the symmetry axis, their grouping and interpretation to generate higher primitives in images. We also intend to apply these tools to the detection of local symmetries.

References

[1] H. Asada and M. Brady. The Curvature Primal Sketch. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:2–14, January 1986.



(a) Intensity Image (b) Parallel Symmetry (c) Positioning



(d) Intensity Image (e) Parallel Symmetry (f) Positioning

Figure 11: Positioning of a Torus

[2] D. H. Ballard and C. M. Brown. *Computer Vision.* Prentice Hall, Englewood Cliff, NJ, 1982.

[3] R. Bartels, J. Beatty, and B. Barsky. *An Introduction to Splines for use in Computer Graphics and Geometric Modeling.* Morgan Kaufmann, Los Altos, CA 94022, 1987.

[4] H. Blum. *A Transformation for Extracting New Descriptors of Shape.* MIT Press, Cambridge, MA, 1967.

[5] J. Brady and H. Asada. Smoothed Local Symmetries and their Implementation. Technical Report 757, Massachusetts Institute of Technology Artificial Intelligence Laboratory, February 1984.

[6] R. Brent. *Algorithms for Minimization without Derivatives.* Prentice Hall, Englewood Cliffs, NJ, 1973.

[7] R. A. Brooks. Symbolic Reasoning among 3-D Models and 2-D Images. *Artificial Intelligence*, (17):285–348, 1981.

[8] J. F. Canny. Finding Edges and Lines in Images. Technical Report 720, Massachusetts Institute of Technology Artificial Intelligence Laboratory, 1983.

[9] J. F. Canny. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[10] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design.* Academic Press, San Diego, CA, 1988.

[11] H. Freeman. On the Encoding of Arbitrary Geometric Configurations. *IRE Transactions on Electronic Computers*, pages 260–268, June 1961.

[12] S. A. Friedberg. Finding Axes of Skewed Symmetry. *Journal of Computer Vision, Graphics, and Image Processing*, 34(2):138–155, 1986.

[13] A. Huertas and R. Nevatia. Detecting Buildings in Aerial Images. *Journal of Computer Vision, Graphics, and Image Processing*, 41(2):131–152, 1988.

[14] T. Kanade. Recovery of the Three-Dimensional Shape of an Object from a Single View. *Artificial Intelligence*, 17:409–460, 1981.

[15] D. Marimont. A Representation for Image Curves. In *Proceedings of the National Conference on Artificial Intelligence*, pages 237–242, Austin, TX, 1984.

[16] G. Medioni and R. Nevatia. Segment-Based Stereo Matching. *Journal of Computer Vision, Graphics, and Image Processing.* (31):2–18, 1985.

[17] G. Medioni and Y. Yasumoto. Corner Detection and Curve Representation using Cubic B-Splines. *Journal of Computer Vision, Graphics, and Image Processing*, (39):267–278, 1987.

[18] R. Mohan. Perceptual Organization for Computer Vision. Ph.D. Dissertation 254, Institute for Robotics and Intelligent Systems, School of Engineering, University of Southern California, Los Angeles, California 90089-0273, August 1989.

[19] J. Mundy, A. Heller, and D. Thompson. The Concept of an Effective Viewpoint. In *Proceedings of the DARPA Image Understanding Workshop*, pages 651–659, Cambridge, MA, 1988.

[20] V. Nalwa. Line-Drawing Interpretation. In *Proceedings of the DARPA Image Understanding Workshop*, pages 956–967, Los Angeles, CA, 1987.

[21] V. Nalwa and E. Pauchon. Edgel-Aggregation and Edge-Description. In *Proceedings of the DARPA Image Understanding Workshop*, pages 176–185, Miami, 1985.

[22] R. Nevatia and K.R. Babu. Linear Feature Extraction and Description. *Journal of Computer Graphics and Image Processing*, 13:257–269, 1980.

[23] R. Nevatia and Binford T. O. Description and Recognition of Complex Curved Objects. *Artificial Intelligence*, 8:77–98, 1977.

[24] T. Pavlidis. *Algorithms for Graphics and Image Processing.* Springer-Verlag, 1982.

[25] W. Perkins. A Model Based Vision System for Industrial Parts. *IEEE Transactions on Computers*, (C27):126–143, February 1978.

[26] P. Perona and J. Malik. Scale Space and Edge Detection using Anisotropic Diffusion. In *Proceedings of IEEE Workshop on Computer Vision*, pages 16–22, Miami, 1987.

[27] M. Plass and M. Stone. Curve-Fitting with Piecewise Parametric Cubics. *ACM Transactions on Computer Graphics*, 17(3):229–239, 1983.

[28] J. Ponce. Ribbons, Symmetries, and Skew Symmetries. In *Proceedings of the DARPA Image Understanding Workshop*, pages 1074–1079, Cambridge, Massachusetts, 1988.

[29] J. Ponce and D. J. Kriegman. On Recognizing and Positionning Curved 3-D Objects from Image Contours. In *Proceedings of the DARPA Image Understanding Workshop*, pages 461–470, Palo Alto, CA, 1989.

[30] K. Rao, R. Nevatia, and G. Medioni. Issues in Shape Description and an Approach for Working with Sparse Data. In *Workshop on Spatial Reasoning and Multi-Sensor Fusion*, pages 168–177, Chicago, October 1987.

[31] A. Rosenfeld. Axial Representations of Shape. *Journal of Computer Vision, Graphics, and Image Processing*, (33):156–173, 1986.

[32] P. Saint-Marc, J.S. Chen, and G. Medioni. Adaptive Smoothing: a General Tool for Early Vision. In *Proceedings of IEEE Computer Vision and Pattern Recognition*, pages 618–624, San Diego, California, 1989.

[33] F. Stein and G. Medioni. Gray-Code Representation and Indexing. Technical report, Institute for Robotics and Intelligent Systems, School of Engineering, USC, Los Angeles, CA 90089-0273, August 1989.

[34] F. Ulupinar and R. Nevatia. Using Symmetries for Analysis of Shape from Contour. In *Proceedings of the International Conference on Computer Vision*, pages 414–426, 1988.

[35] S. Y. K. Yuen. Shape from Contour Using Symmetries. Technical Report 141, School of Cognitive Sciences, University of Sussex, Falmer, Brighton BN1 9QN, 1989.

B-snakes: implementation and application to stereo*

Sylvie Menet†, Philippe Saint-Marc‡ and Gérard Medioni
Institute for Robotics and Intelligent Systems
Powell Hall 204
University of Southern California
Los Angeles, California 90089-0273

Abstract

We present B-snakes: a new implementation of snakes using parametric B-splines. This active contour model exhibits advantages of B-splines: compact representation, local control and the possibility to include corners. This implementation is significantly faster without loss of generality. Experiments on delineation of building roofs in stereo aerial images are also presented.

1 INTRODUCTION

Real-world images are often noisy and too complex to expect local, low level operations to perform a complete analysis. Higher level features have to be derived and used in order to get a better delineation of objects.

When there exist enough constraints, it is possible to use deformable models, which adapt to the data, an example being "snakes" [Kass *et al.*, 1988].

We present an implementation of such models based on parametric B-spline approximation, which offers many advantages. Among them, it provides a compact local representation of a curve, in terms of its control-points. Furthermore, B-splines have the ability to represent corners, that is, to locally override smoothness constraints. A new active contour model is built using this B-spline approximation for a curve and is called a "B-snake". These B-snakes converge much faster than snakes and can include corners without invoking specific models.

As an application, B-snakes are used to precisely outline the boundaries of building roofs in stereo pairs of urban scenes, given an initial rough outline from a standard stereo matching algorithm [Cochran and Medioni, 1989].

*This research was supported in part by DARPA contract F33615-C-1436

†Supported by a grant from the French Direction Générale de l'Armement, contract ERE 89/1460/DRET/DS/SR. Address: ONERA, 29 Ave de la Div Leclerc, 92320 Chatillon/Bagneux, France.

‡Permanent address: Matra-SEP Image et Informatique, Signal and Image Processing Laboratory, BP 235 "Les Miroirs", 38 Bd Paul Cézanne, Guillancourt, 78052 St-Quentin en Yvelynes Cédex, FRANCE

The paper is organized as follows: we briefly review snakes and their applications, then give details of our B-snake implementation, and illustrate the methodology on the accurate delineation of building tops in stereo pairs of urban scenes.

2 SNAKES

A snake is a deformable continuous curve, whose shape is controlled by internal forces (the implicit model) and external forces (the data). Internal forces act as a smoothness constraint, and external forces guide the active contour towards image features.

Let $v(s) = (x(s), y(s))$ be the parametric description of the snake ($s \in [0,1]$). Its total energy can be written as:

$$\begin{aligned} E_{snake} &= \int_0^1 E_s(v(s))ds \\ &= \int_0^1 [E_{int}(v(s)) + E_{ext}(v(s))]ds \end{aligned} \quad (1)$$

with:

$$E_{int}(s) = \frac{1}{2}(\alpha(s) \mid v_s(s) \mid^2 + \beta(s) \mid v_{ss}(s) \mid^2) \quad (2)$$

The goal is to find the snake that minimizes equation (1), given some external energy adapted to image features to extract ($E_{edge} = - \mid \nabla I(x,y) \mid^2$, for example) and internal energy whose expression is given by (2). The first order term makes the snake act like a membrane and the second order one like a thin plate. This energy is the regularizing term of the minimization.

The minimization of (1) is solved by using the calculus of variations and resolving Euler equations, and yields the following equations in the discrete case [Kass *et al.*, 1988]:

$$\begin{cases} Ax + F_x(x,y) = 0 \\ Ay + F_y(x,y) = 0 \end{cases} \quad (3)$$

where $F = E_{ext}$ depends on the image features to extract and A is a pentadiagonal matrix depending on α and β.

This system of equations in (x,y) is solved by introducing an energy dissipation functional to dissipate the kinetic energy during the motion. Let γ be the Euler step size. The expression of the snake as a function of time is then:

$$\begin{cases} x_{t+1} = (A + \gamma I)^{-1}(\gamma x_t - F_x(x_t, y_t)) \\ y_{t+1} = (A + \gamma I)^{-1}(\gamma y_t - F_y(x_t, y_t)) \end{cases} \quad (4)$$

$(A + \gamma I)^{-1}$ can be calculated by LU decompositions in $O(n)$ time (with n being the length of the snake).

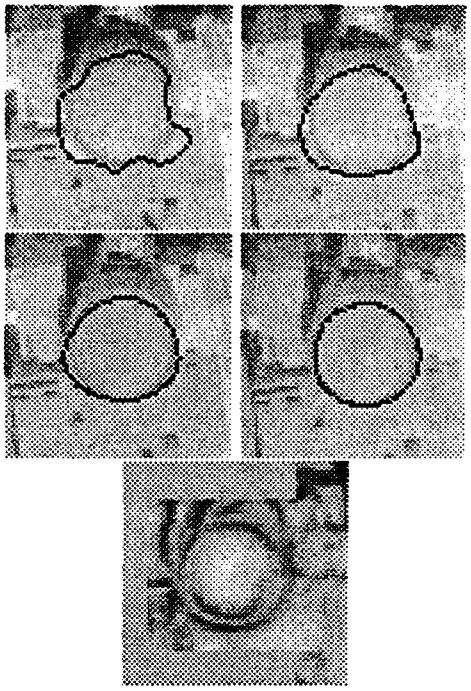


Figure 1: Example of snake convergence, the external energy is the negated gradient.

Figure 2 shows an example of convergence.

This active contour model fits in an interactive human-machine environment when the user supplies an initial estimate of the object to extract and the snake is used to refine the results [Kass *et al.*, 1988; Fua and Hanson, 1989b; Fua and Hanson, 1989a; Fua and Leclerc, 1988]. However, it is also useful in an automatic processes when a first estimate is given by a first processing level [Ferrie *et al.*, 1989; Zucker *et al.*, 1988].

This tool has been applied in motion [Kass *et al.*, 1988], in stereo matching [Kass *et al.*, 1988; Fua and Leclerc, 1988], and, more generally, it can be used to match a deformable model to an image by means of energy minimization.

Different implementations have been performed, for example, Fua [Fua and Hanson, 1989a] uses a tool from information theory: he minimizes an objective function that is the length of encoding the result. This methodology is general and applies to object recognition using generic models. Amini [Amini *et al.*, 1988] uses dynamic programming to minimize the energy, and can handle hard local constraints. Berger [Berger, 1990] allows the snake to grow along features, and also to break.

Unfortunately, the convergence rate of a snake, using all points, is rather slow. Hence, some researchers [Fua and Leclerc, 1988; Amini *et al.*, 1988] use a polygonal approximation of the curve, but then smoothness can no longer be guaranteed. Another problem is that the only way to include corners is to set $\alpha = 0$ at some locations, so the "cornerness" of the curve is not implicit.

A better way to simultaneously solve these problems is to use a parametric B-spline approximation of curves [Bartels *et al.*, 1987], as the next section shows. We call this new model a "B-snake".

3 B-SNAKES

In this model, the curve is replaced by its approximation by a B-spline and the energy of the approximation is minimized.

We first discuss the advantages of the scheme, then explain how to compute the B-spline approximation of the curve, and finally show the minimization procedure with B-snakes.

Let u be the parameter describing the approximating curve (we take u instead of s to remain consistent with notations in [Bartels *et al.*, 1987]), and $Q(u) = (x(u), y(u))$.

In this approximation, the curve is split into segments, and the joints between adjacent curve segments are called *knots*. Each curve segment is approximated by a piecewise polynomial function (order k), which is obtained by a linear combination of basis functions B_i and a set of *control vertices* $V_i = (X_i, Y_i)$:

$$Q(u) = \sum_{i=0}^{i=m} V_i B_i(u) \qquad (5)$$

The control polygon can be calculated by performing a least-square fit of the data by the B-spline curve (paragraph 3.2).

In the following, let $p+1$ be the number of points of the curve and $m+1$ the number of vertices of the control polygon.

As is shown in paragraph (3.3), substituting v by $Q(u)$ in the snake energy equation (1) yields a similar system to (4), whose unknowns are the control vertices and therefore whose size is only $m+1$ instead of $p+1$.

3.1 Advantages of approximating B-splines

Local control : elementary B-splines B_i have local support, so that modifying the position of a data-point causes only a small part of the curve to change.

Continuity control : B-splines are defined with continuity properties at each point: order k B-splines are C^{k-2} continuous. But it is possible to control the continuity at the knots, by accepting multiple knots. These are obtained by letting successive knots be equal, which causes intermediate intervals to be empty. Let μ be the multiplicity degree of a knot, the continuity at this knot is then: $C^{k-1-\mu}$. When μ is equal to $k-1$ the knot is C^0 continuous and the corresponding control point is interpolated by the curve.

This property is very interesting for the B-snake model: if we introduce a multiple knot whose degree of multiplicity is equal to $k-1$, the first and second derivatives are no longer continuous at this

knot and the smoothness constraint is broken: a corner appears at this knot.

3.2 Control polygon

To find the control polygon at time 0, we perform a least-squares fit of the data by a B-spline curve [Bartels *et al.*, 1987; Saint-Marc and Medioni, 1990].

We want to minimize the distance between the discrete data points of the original curve and its approximation by a B-spline. This distance is given by the expression:

$$R = \sum_{j=0}^{p} \mid Q(u_j) - P_j \mid^2 = \sum_{j=0}^{p} ((x(u_j)-x_j)^2 + (y(u_j)-y_j)^2)$$

where u_j is some parameter value associated with the j^{th} data point, and $Q(u_j)$ is given by:

$$Q(u) = \sum_{i=0}^{i=m} V_i B_i(u) = \sum_{i=0}^{i=m} (X_i B_i(u), Y_i B_i(u)) \qquad (6)$$

Since this equation is quadratic, the minima occurs for those values of X_l and Y_l such that:

$$\begin{cases} \frac{\partial R}{\partial X_l} &= 0 \\ \frac{\partial R}{\partial Y_l} &= 0 \end{cases}$$

where l ranges between 0 and m. So, we obtain:

$$\begin{aligned} \sum_{i=0}^{m} X_i \sum_{j=0}^{p} B_i(u_j) B_l(u_j) &= \sum_{j=0}^{p} x_j B_l(u_j) \\ \sum_{i=0}^{m} Y_i \sum_{j=0}^{p} B_i(u_j) B_l(u_j) &= \sum_{j=0}^{p} y_j B_l(u_j) \end{aligned} \qquad (7)$$

This equation can be solved by a LU decomposition.

The choice of the number of vertices, $m+1$, determines how close to the original data the approximation is, which is measured by R. An automatic choice can then be performed [Saint-Marc and Medioni, 1990]: we set a fitting tolerance r_0 and we find the value of $m+1$ which yields the normalized distance $r = R/(p+1)$ closer to r_0, using a binary search approach.

3.3 Minimization resolution

We want to minimize equation (1) by substituting the curve v by the analytical expression of its B-spline approximation (6).

The total energy of the curve is then:

$$\begin{aligned} E = \sum_{j=0}^{p} \{ & \frac{1}{2}\alpha(u_j)[(\sum_{i=0}^{m} X_i B_i'(u_j))^2 + (\sum_{i=0}^{m} Y_i B_i'(u_j))^2] \\ & + \frac{1}{2}\beta(u_j)[(\sum_{i=0}^{m} X_i B_i''(u_j))^2 + (\sum_{i=0}^{m} Y_i B_i''(u_j))^2] \\ & + F(v(u_j))\} \end{aligned} \qquad (8)$$

We are looking for control points coordinates X_i, Y_i that minimize E, that is, that satisfy:

$$\forall l \in \{0, \cdots, m\} \begin{cases} \frac{\partial E}{\partial X_l} = 0 \\ \frac{\partial E}{\partial Y_l} = 0 \end{cases}$$

That yields for the X coordinate:

$$\begin{aligned} \sum_{j=0}^{p} \; & [\alpha(u_j) B_l'(u_j) \sum_{i=0}^{m} X_i B_i'(u_j) + \\ & \beta(u_j) B_l''(u_j) \sum_{i=0}^{m} X_i B_i''(u_j) + \\ & B_l(u_j) \frac{\partial}{\partial x} F(\sum_{i=0}^{m} X_i B_i(u_j), \sum_{i=0}^{m} Y_i B_i(u_j))] \\ & = 0 \end{aligned} \qquad (9)$$

and a similar equation for Y. When we change the summation order, we get:

$$\begin{aligned} \sum_{i=0}^{m} X_i \; & [\sum_{j=0}^{p} \alpha(u_j) B_l'(u_j) B_i'(u_j) + \\ & \sum_{j=0}^{p} \beta(u_j) B_l''(u_j) B_i''(u_j)] \\ + \; & \sum_{j=0}^{p} B_l(u_j) F_x(v(u_j)) = 0 \end{aligned}$$

for l ranges from 0 to m.

This equation set can be written in the same matrix form as (3), with $m+1$ equations of $m+1$ unknowns (X, Y) instead of $p+1$ (x, y):

$$\begin{cases} A_b X + G_x(x,y) &= 0 \\ A_b Y + G_y(x,y) &= 0 \end{cases} \qquad (10)$$

where A_b is still a band matrix.

This system can be solved in way similar to the original snakes (4), and we have:

$$\begin{cases} X_{t+1} &= (A_b + \gamma I)^{-1}(\gamma X_t - G_x(x_t, y_t)) \\ Y_{t+1} &= (A_b + \gamma I)^{-1}(\gamma Y_t - G_y(x_t, y_t)) \end{cases} \qquad (11)$$

4 APPLICATION : BUILDING TOPS DELINEATION FROM STEREO DATA

The detection of cultural features, such as roads and buildings in aerial imagery is an important application area in Computer Vision.

In recent work Fua [Fua and Hanson, 1989a; Fua and Leclerc, 1988] has proposed to detect such buildings by refining a coarse estimate through a parameter estimation phase. Mohan [Mohan and Nevatia, 1988] defines a building as a collation of rectangles and proposes to solve the selection process by a Constraint Satisfaction Network.

These methods use monocular information only, such as edges, to generate and verify hypotheses. When stereo data is available, they use it mostly in the verification stage to refine the estimates.

Here, we propose instead to use stereo first to guide in the detection of elevated structures, on the basis that their disparity is bound to be different from the disparity of the background, and to refine the estimates using monocular information.

Most stereo algorithms (see [Barnard and Fischler, 1982; Dhond and Aggarwal, 1989] for surveys) produce reliable results in images of rolling terrain, but degrade ungracefully when depth discontinuities occur, since the smoothness assumption becomes violated. This is true for area-based and feature-based methods. We use here an algorithm which combines both approaches, as described in [Cochran and Medioni, 1989]. The buildings roofs appear as regions of constant disparity, but their boundaries are very approximate, generally ragged.

We can refine them by using:

- monocular information: buildings are likely to generate intensity edges;
- smoothness: building boundaries are mostly smooth, with the exception of some corners;
- invariance: the boundaries should correspond in both images.

To turn these observations into a computational framework, we use the B-snakes described above. The internal energy captures the smoothness constraint, and we define an appropriate external energy for the other two constraints, as shown below.

To solve the problem introduced by corners, we proceed in two stages: first, the boundary is supposed smooth, and the snake reaches its convergence state, then potential corners are detected as extrema of curvature and the B-snake model is applied again.

We now give the details of the process and present some illustrative results.

4.1 Stereo energy

Kass [Kass *et al.*, 1988] applies snakes to the problem of stereo matching. According to some psychological evidence [Burt and Julesz, 1980], he assumes that, if two contours correspond then the disparity varies slowly along the 3-D contour. This constraint can be expressed in an additional energy functionnal:

$$E_{stereo} = (v^L(s) - v^R(s))^2$$

where v^L and v^R are left and right snake contours.

Fua [Fua and Hanson, 1989a] uses a stereographic effectiveness term which encodes the projected patch in the second image, while knowing its photometry in the first.

In our approach, the contours of non-nul disparity areas are the first estimate of objects contours we want to improve, that is, the initialization of the snakes at time 0.

Furthermore, we can combine the left and right external energy of each object, by projecting the right one on the left one through the disparity map (equation 12). This allows us to filter non matching areas and to reinforce constraints in matched areas.

$$E_{stereo}(s) = E_L(s) + d(s)E_R(s) \qquad (12)$$

Since edges are likely to correspond to depth or surface orientation discontinuities, we use edge information as monocular external energy. This energy supplies the feature-based information often used in stereo matching algorithm but which yields a sparse disparity map.

In order to increase the efficiency when the snake is too far from the edges, a distance map such as Chamfer distance [Barrow *et al.*, 1977] is added to the edge information.

4.2 Discontinuities C^0

Polygonal objects can be processed without a priori knowledge on their shape, by using a method in two steps:

1. First stage: Regular B-snakes are implemented and their energy is minimized until equilibrium.
2. Second stage: Corners are detected at points of maxima curvature (equation 13) and new B-snakes are implemented with multiple-knots at the corners. Then B-snakes converge from their previous state toward a new equilibrium.

$$\rho = \frac{x_u y_{uu} - x_{uu} y_u}{(x_u^2 + y_u^2)^{\frac{3}{2}}} \qquad (13)$$

Images 2 and 3 show this process.

When corners are detected, we assume in this application, that polygonal objects are encountered. Then, new parameters are used in the second stage, to emphasize the behavior of the B-snake acting as a strong rod between corners.

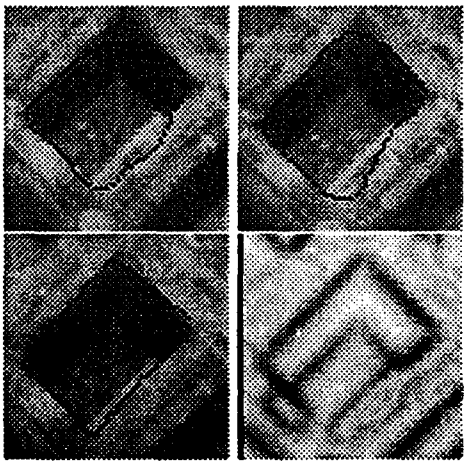


Figure 2: First example: Initialization, result of first step and final result. The external energy is also shown.

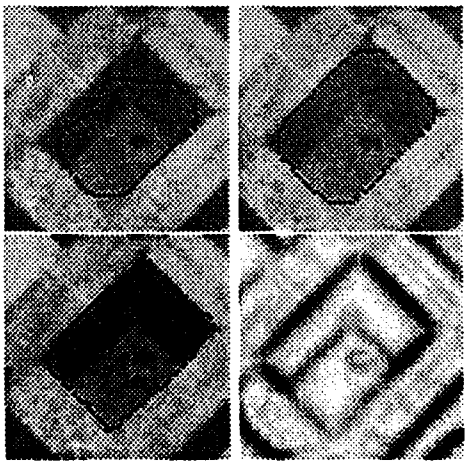


Figure 3: Second example: Initialization, result of first step and final result. The external energy is also shown.

4.3 Results

Images 4 to 17 show two series of examples obtained with quadratic B-snakes.

For each example, the left and right disparity maps are shown, which are blurred and noisy at corners. The initialization of B-snakes are extracted from the left one, and the process is performed on this side.

Results are shown at both steps of the process described above. We can see that after the first step, roofs borders are improved, but remained rounded at corners. They become sharper after the second step.

While it gives the direction of the nearest edge, the Chamfer distance helps the convergence especially when the curve is too far from the edges. But it does not provide reliable information at locations of multiple nearby edges. This and the lack of edge information at some locations contribute to cause B-snakes to stabilize into local minima.

Furthermore, this energy makes the B-snakes to shrink or to expand only if the first estimate is around local maxima otherwise, it shrinks until vanishing (for example: the highest tower cannot be handled considering the poor edge information used).

5 CONCLUSION

Snakes provide a tool to solve many vision problems by means of global energy-minimizing, while taking into account geometrical model of curves and image features information. As the energy is integrated along the entire length of the curve, it is less sensitive to image noise and various photometric anomalies.

We have improved this tool by using parametric B-spline approximations of curves that yield increasing convergence speed and allow the so-called *B-snake* to include corners.

Then, the B-snake can be applied to adjustment of non-smooth shapes. For example, it is able to refine the delineation of building tops from stereo aerial images, with a good accuracy, without using a priori knowledge or generic model.

Acknowledgements

We would like to thank J.L. Jezouin from Matra-SEP for providing the original stereo pairs used in this study.

References

[Amini *et al.*, 1988] A. Amini, S. Tehrani, and T. Weymouth. Using dynamic programming for minimizing the energy of active contours in the presence of hard constraints. In *Proceedings of 2nd International Conference on Computer Vision*, pages 95–99, Tampa, Florida, 1988.

[Barnard and Fischler, 1982] S. Barnard and M. Fischler. Computational stereo. *ACM Computing Surveys*, 14(4):553–572, December 1982.

[Barrow *et al.*, 1977] H. Barrow, J. Tenebaum, R. C. Bolles, and H. C. Wolf. Parametric correspondence and chamfer matching. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 659–663, Cambridge, Massachusetts, August 1977.

[Bartels *et al.*, 1987] R. Bartels, J. Beatty, and B. Barsky. *An Introduction to Splines for use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann, Los Altos, CA 94022, 1987.

[Berger, 1990] M. O. Berger. Snake growing. In *First European Conference on Computer Vision*, pages 570–572, Antibes, France, April 1990.

[Burt and Julesz, 1980] P. Burt and B. Julesz. A disparity gradient limit for binocular fusion. *Science*, 208:615–617, 1980.

[Cochran and Medioni, 1989] S. D. Cochran and G. Medioni. Accurate surface description from binocular stereo. In *Proceedings of Workshop on Interpretation of 3D Scenes*, pages 16–23, Austin, Texas, Nov. 1989.

[Dhond and Aggarwal, 1989] U. R. Dhond and J. K. Aggarwal. Structure from stereo-A review. *IEEE Transactions on Systems, Man & Cybernetics*, 19(6):1489–1510, November/December 1989.

[Ferrie *et al.*, 1989] F. Ferrie, J. Lagarde, and P. Whaite. Darboux frames, snakes, and super-quadrics: Geometry from the bottom-up. In *Proceedings of Workshop on Interpretation of 3D Scenes*, pages 170–176, Austin, Texas, Nov. 1989.

[Fua and Hanson, 1989a] P. Fua and A.J. Hanson. Objective function for feature discrimination theory. In *Proceedings of the DARPA Image Understanding Workshop*, pages 443–460, May 1989.

[Fua and Hanson, 1989b] P. Fua and A.J. Hanson. An optimisation framework of feature extraction: Applications to semiautomated and automated feature extraction. In *Proceedings of the DARPA Image Understanding Workshop*, pages 676–694, May 1989.

[Fua and Leclerc, 1988] P. Fua and Y. G. Leclerc. Model driven edge detection. In *Proceedings of the DARPA Image Understanding Workshop*, volume 2, pages 1016–1021, Cambridge, Massachusetts, April 1988.

[Kass *et al.*, 1988] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active Contour Models. *International Journal of Computer Vision*, 1:321–331, January 1988.

[Mohan and Nevatia, 1988] R. Mohan and R. Nevatia. Perceptual grouping for the detection and description of structures in aerial images. In *Proceedings of the DARPA Image Understanding Workshop*, pages 512–526, Boston, Massachusetts, April 1988.

[Saint-Marc and Medioni, 1990] P. Saint-Marc and G. Medioni. B-spline contour representation and symmetry detection. In *First European Conference on Computer Vision*, pages 604–606, Antibes, France, April 1990.

[Zucker *et al.*, 1988] S. Zucker, C. David, A. Dobbins, and L. Iverson. The organization of curve detection: Coarse tangent fields and fine spline coverings. In *Proceedings of 2nd International Conference on Computer Vision*, pages 568–577, Tampa, Florida, Dec. 1988.

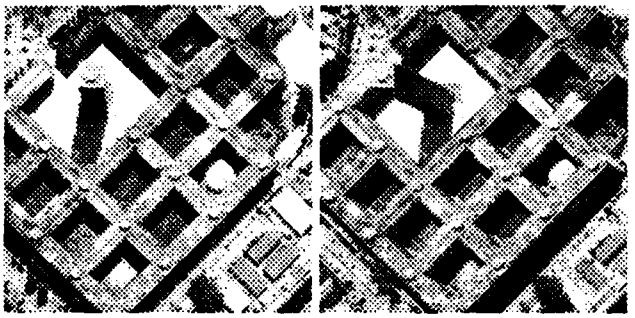


Figure 4: First example: stereo intensity images

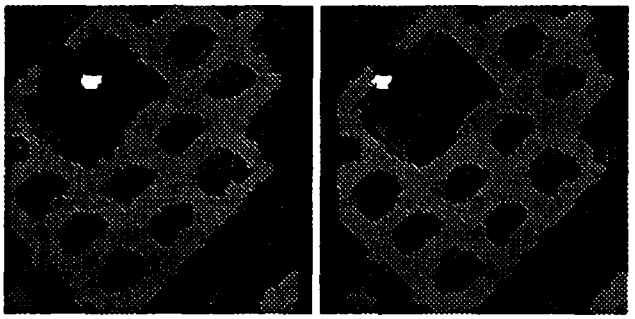


Figure 5: Left and right disparity map

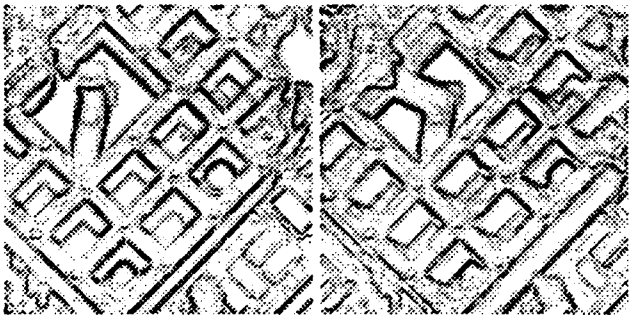


Figure 6: Left and right negated gradient

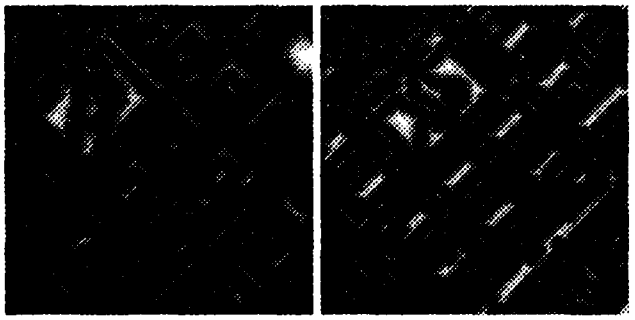


Figure 7: Left and right Chamfer distance

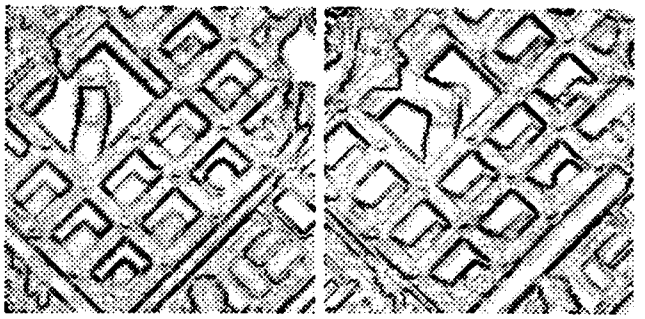


Figure 8: Left and right final energies

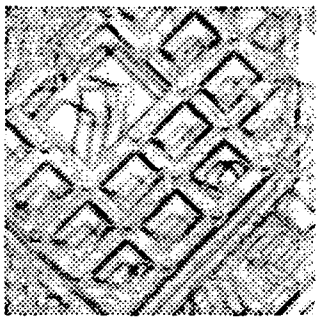


Figure 9: Global stereo energy

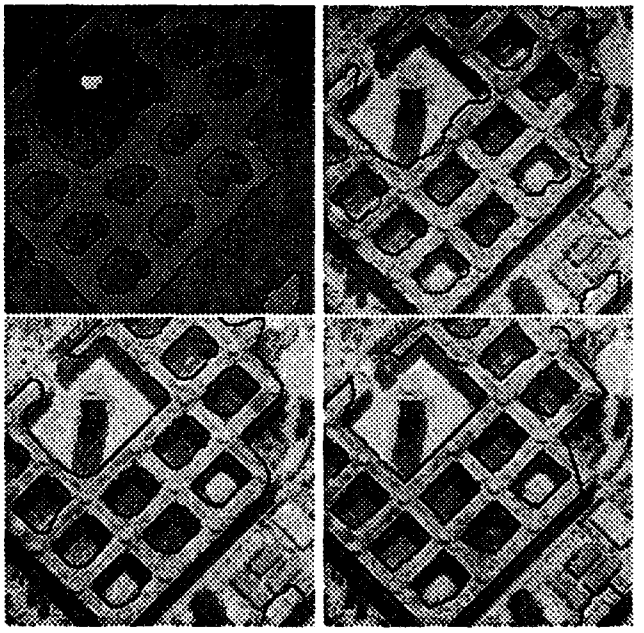


Figure 10: Different steps of delineation of buildings roofs from the first estimate of B-snakes from edges of disparity map to the final result with corners

Figure 11: Second example: stereo intensity images

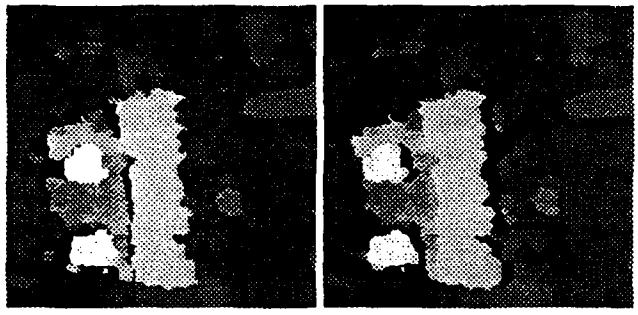


Figure 12: Left and right disparity

Figure 13: Left and right negated gradient

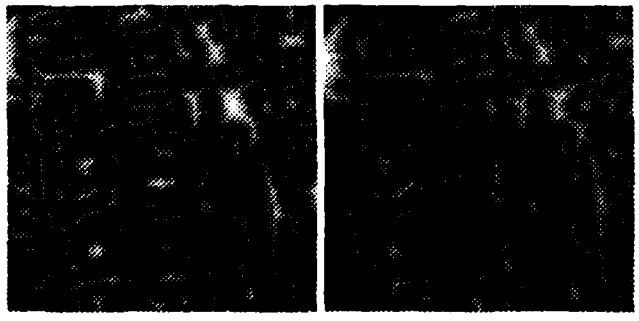


Figure 14: Left and right Chamfer distance

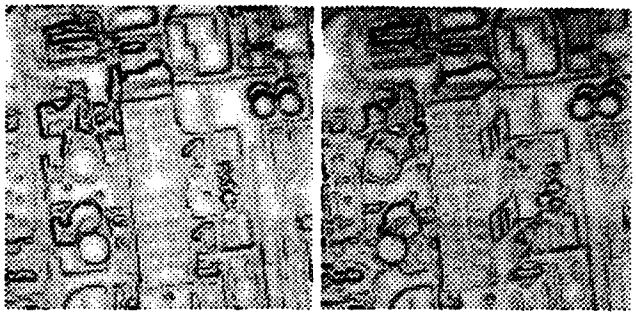


Figure 15: Left and right global energy

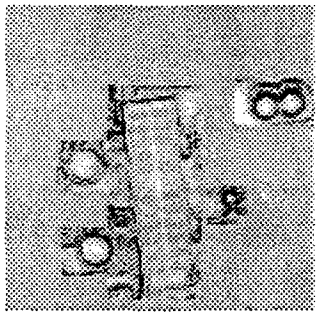


Figure 16: Global stereo energy

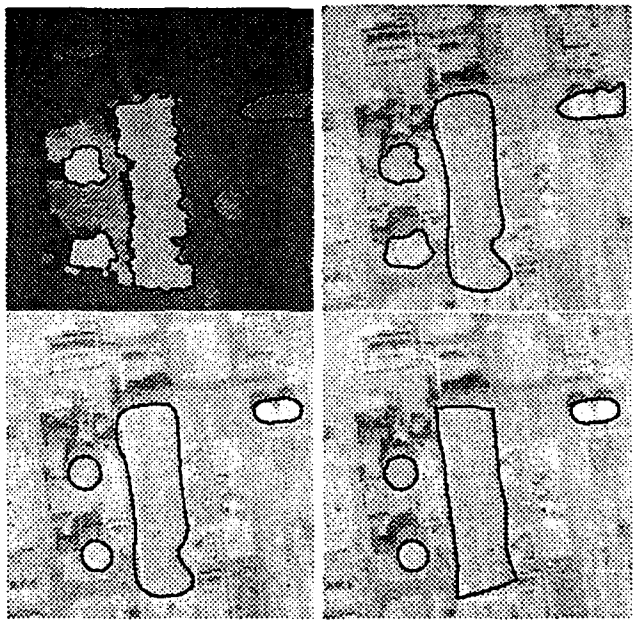


Figure 17: Different steps of delineation of buildings roofs from the first estimate of B-snakes from edges of disparity map to the final result with corners